NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

USING MODEL BASED SYSTEMS ENGINEERING AND THE SYSTEMS MODELING LANGUAGE TO DEVELOP SPACE MISSION AREA ARCHITECTURES

by

Dustin B. Jepperson

September 2013

Thesis Advisor:	Mark Rhoades
Second Reader:	Kristin Giammarco

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2013	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE USING MODEL BASED SYSTEMS ENGINEERING AND THE SYSTEMS MODELING LANGUAGE TO DEVELOP SPACE MISSION AREA ARCHITECTURES		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Dustin B. Jepperson			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) Los Angeles Air Force Base – Space and Missiles Systems Center 483 North Aviation Blvd El Segundo CA 90245		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. government. IRB protocol number NPS.2013.0069 .			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release;distribution is unlimited		**12b. DISTRIBUTION CODE** A	

13. ABSTRACT (maximum 200 words)

Model based systems engineering (MBSE) is explored as an alternative to the Department of Defense (DoD)'s heavily document-driven processes for architecture development and acquisition management. MBSE can be employed to meet the standards set in the DoD acquisition framework. Data exchange specifications, such as the application protocol 233 (AP233), can be implemented to enable synergistic benefits to data analysis across the enterprise. Architecture development techniques, including the structured analysis and design technique and the systems modeling language (SysML), are introduced to aid in the development and assessment of space system mission area architectures, enabling rigorous mathematical analysis to support key programmatic decisions. A detailed example of the application of SysML, in conjunction with MBSE principles, is provided for the Overhead Persistent Infrared mission area, specifically the Space Based Infrared Surveillance System. A three-phase adoption approach is recommended: first identify, list, and manage the configuration of all critical program models, processes, and tools used throughout the DoD. Second, mandate a data exchange specification, such as the International Organization for Standardization (10303 AP233 standard, across the DoD space acquisition community. Finally, further standardize the implementation of MBSE practices through implementation of SysML. Heuristics for developing system architecture are provided.

14. SUBJECT TERMS Model Based Systems Engineering (MBSE), Systems Modeling Language (SySML), Structured Analysis and Design Technique (SADT), Application Protocol 233 (AP233), Department of Defense Architecture Framework (DoDAF), Space Mission Area System Architecture (MASA), Overhead Persistent Infrared (OPIR), Space Based Infrared Surveillance System (SBIRS), System Architecture Heuristics	**15. NUMBER OF PAGES** 147
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

USING MODEL BASED SYSTEMS ENGINEERING AND THE SYSTEMS MODELING LANGUAGE TO DEVELOP SPACE MISSION AREA ARCHITECTURES

Dustin B. Jepperson
Captain, United States Air Force
B.S., University of Wisconsin, Madison, 2008

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
September 2013

Author: Dustin B. Jepperson

Approved by: Mark Rhoades, PhD
Thesis Advisor

Kristin Giammarco, PhD
Second Reader

Clifford Whitcomb, PhD
Chair, Department of Systems Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Model based systems engineering (MBSE) is explored as an alternative to the Department of Defense (DoD)'s heavily document-driven processes for architecture development and acquisition management. MBSE can be employed to meet the standards set in the DoD acquisition framework. Data exchange specifications, such as the application protocol 233 (AP233), can be implemented to enable synergistic benefits to data analysis across the enterprise. Architecture development techniques, including the structured analysis and design technique and the systems modeling language (SysML), are introduced to aid in the development and assessment of space system mission area architectures, enabling rigorous mathematical analysis to support key programmatic decisions. A detailed example of the application of SysML, in conjunction with MBSE principles, is provided for the Overhead Persistent Infrared mission area, specifically the Space Based Infrared Surveillance System. A three-phase adoption approach is recommended: first identify, list, and manage the configuration of all critical program models, processes, and tools used throughout the DoD. Second, mandate a data exchange specification, such as the International Organization for Standardization (10303 AP233 standard, across the DoD space acquisition community. Finally, further standardize the implementation of MBSE practices through implementation of SysML. Heuristics for developing system architecture are provided.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

Alt	Alternative
AoA	Analysis of Alternatives
AP233	Application Protocol 233
CBAs	Capability Based Assessments
CCaR	Comprehensive Cost and Requirement System
CDD	Capabilities Development Document
CONOPS	Concept of Operations
CPD	Capabilities Production Document
DoD	Department of Defense
DoDAF	Department of Defense Architecture Framework
DSP	Defense Support Program
FFRDC	Federally Funded Research and Development Center
GEO	Geosynchronous Orbit
HEO	Highly Elliptical Orbit
I/O	Input/Output
ICD	Initial Capabilities Document
IDEF0	Integrated DEFinition zero
INCOSE	International Council on Systems Engineering
ISO	International Organization for Standardization
LAAFB	Los Angeles Air Force Base
MASA	Mission Area System Architecture
MBSE	Model Based Systems Engineering
MDA	Model Driven Architecture
MOE	Measure of Effectiveness
OMB	Office of Management and Budget
OMG	Object Management Group
OPIR	Overhead Persistent InfraRed
OV-1	Operational View—1
PIM	Program Independent Model
PLCM	Product Life Cycle Management

POR	Program of Record
PSMs	Platform Specific Models
RFP	Request for Proposal
SADT	Structured Analysis and Design Technique
SBIRS	Space Based InfraRed System
SDS	System Design Specification
SE	Systems Engineering
SEC	Systems Engineering Capability
SEC-ARCH	Systems Engineering Capability—Architecture
SEC-ARCH	Systems Engineering Capability—Architecture
SEC-Total	Systems Engineering Capability—Total
SEC-TRD	Systems Engineering Capability—Trade Studies
SEH	Systems Engineering Handbook
SEP	Systems Engineering Plan
SMART	System Metric and Reporting Tool
SMC	Space and Missiles Systems Center
SoS	Systems-of-Systems
SysML	Systems Modeling Language
TEMP	Test and Evaluation Master Plan
UML	Unified Modeling Language
USG	United States Government
WBS	Work Breakdown Structure
XMI	Extensible Markup Language (XML) Metadata Interchange

EXECUTIVE SUMMARY

The objective of this thesis is to explore the potential benefits of using a model based systems engineering (MBSE) approach, facilitated by a structured architecture modeling language such as the Systems Modeling Language (SysML), to develop and employ mission area architectures for Department of Defense (DoD) space systems. Recently, the need to capture and develop comprehensive architectures for space mission areas within the DoD has drastically increased. It is proposed that in order to respond to this challenge, it is recommended that the DoD depart from its exclusive use of document-driven processes for architecture and acquisition management and adopt a rigorous technique such as MBSE.

MBSE is a formalized approach to modeling and architecting a system across the full system lifecycle. MBSE can be employed to the standards set by the DoD Architecture Framework (DoDAF), which provides the baseline structure and common data meta-model specifications to develop mission area architectures for the DoD, including space systems. Data exchange specifications, such as the ISO 10303 Application Protocol—233 (AP233) standard, can be implemented across a DoD organization to standardize the exchange of architecture and system data between otherwise stove-piped organizational components, enabling synergistic benefits to data analysis across the enterprise. This thesis explores structured techniques, applications, and languages that can be used to enable and aid in the development and assessment of detailed space system architecture, capturing the detailed interactions and interdependencies within and throughout a system and enabling rigorous mathematical analysis to support key programmatic decisions and needs, including the Structured Analysis and Design Technique and the SysML.

In order to realize the maximum benefits of MBSE including, enhanced communications, reduced development risk, improved quality, increased productivity, and enhanced knowledge transfer, a structured architecture development technique such as the Structured Analysis and Design Technique (SADT) or SysML must be implemented across the DoD space community and used to develop space based mission

area architectures. A detailed example of the application of SysML, in conjunction with MBSE principles, is provided for the Overhead Persistent Infrared (OPIR) mission area, and specifically modeled for the Space Based Infrared Surveillance System (SBIRS). This SysML model, once complete and specified with mathematical relationships, can be used to support rigorous engineering analysis. Powerful cost-effectiveness comparisons can then be generated as part of an analysis of alternatives or trade study to inform decision makers by answering the question: How well does any particular architecture satisfy the mission requirements? Ultimately, the overall quality of a system acquisition effort, or project, can be greatly improved through the application of MBSE architecture, modeling and simulation, and trade study activities—all enabled by the development of architecture using SysML.

Adopting MBSE and SysML for the design of DoD space systems will require a fundamental paradigm shift in how the DoD does business, transitioning from what is now a purely document-driven approach. Many potential barriers and limitations exist that may limit or impede the introduction of MBSE practices and the application of SysML. Therefore, a three phase approach is recommended in this thesis. The first incremental phase is to identify, list, and manage the configuration of all critical program models, processes, and tools used throughout the Space and Missile Systems Center (SMC). The second recommended phase is to mandate a data exchange specification, such as the AP233 standard, across the DoD space acquisition community to realize some enterprise benefits and aid in the development of requirements for a more integrated and structured approach such as SysML. Simply implementing a data exchange specification would not fundamentally improve how information is managed at the component level, however. Therefore, the third recommended phase is to further standardize the implementation of MBSE practices by enforcing common processes, standards, models, tools, and techniques across the community. As discussed within this paper, the SysML modeling language is uniquely suited to meet this demand. It is clear that such a paradigm shift is required if the DoD and SMC are to meet their requirements for greater interoperability and ultimately deliver more successful systems through more effective architecture development and acquisition efforts.

ACKNOWLEDGMENTS

I would like to thank my advisors, Mark Rhoades, Kristin Giammarco, and Mary Vizzini for all of their valuable feedback and support of my research and writing.

Of course, none of this would have been possible if not for the incredible support of my loving wife, Laura, and the patience and never-ending inspiration provided by my daughter, Chloe, and son, Rowan. And to my loving parents, Bruce and Rebecca, for instilling in me the motivation to excel and never give up. For it is my family that provides the stimulus that motivates everything I do.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. PROBLEM STATEMENT AND OBJECTIVE

The Space and Missile Systems Center (SMC), Los Angeles Air Force Base (LAAFB), California, is the Department of Defense's (DoD) product center for the Acquisition of all DoD space systems. SMC is organized into program offices responsible for the program management and acquisition of specific space systems, constellations, or portfolios of space systems, along with staff organizations that provide oversight and guidance for all SMC program offices. Given the wide variety of space systems being acquired by SMC and the fact that space systems are among the most complex systems in existence, a great number of elaborate systems engineering and program management tools and processes must be used by every organizational level of SMC and its partners, including the prime and sub-contractors working with the program office to design and build the systems.

Recently, the need to capture and develop comprehensive enterprise architectures for space related mission areas within the DoD has drastically increased. This need is driven by several factors, including the ever increasing complexity of space-based systems-of-systems (SoS) that demand seamless coordination and operation across many organizations and technical interfaces and an austere budget environment demanding that space systems realize maximum efficiencies in the areas of cost, schedule, and performance. A common method is needed to aid in developing and capturing enterprise mission area architectures for use across the DoD space SoS enterprise and applying these common architecture development techniques to all systems designed to operate within the DoD space mission areas.

Typically, a great deal of effort is put forward within each DoD space mission area to plan and develop new systems. The enterprise architectures describing these systems are oftentimes overlooked or over-simplified, only to be later developed in detail by future system acquisition efforts or out of necessity by the defense contractor(s) who is (are) selected to develop and operate the system(s). For instance, although overhead

infrared space systems have been in operations for over 40 years, enterprise architecture descriptions and characterizations for the Overhead Persistent Infrared (OPIR) mission area are just now being developed and shared across the mission area's stakeholders. Prior to recent efforts to capture enterprise mission area architectures, defense contractors selected to develop and operate the legacy OPIR systems would individually develop, build, operate, and maintain system level architectures for their system(s). This system-specific architecture process has been all too common across the DoD space system enterprise for all mission areas, not just OPIR. As a result, the individual system architectures within a particular space related mission area are tied to specific systems within the mission area, while an enterprise architecture capturing the aggregate of all such systems and their related interfaces and inter-dependencies does not exist. Furthermore, these individual system architectures are oftentimes developed and maintained using different (sometimes unique or highly customized) applications, processes, methods, and techniques.

As a result of this disconnect between system level and enterprise space mission area architectures, it is difficult for a program office to capture the true enterprise mission area architecture containing all related systems, therefore making the program office's job—that of planning and developing new future systems to fit within an existing enterprise mission area architecture—quite challenging. In the case of the OPIR mission area, legacy systems such as the Defense Space Program (DSP) and other Intelligence Community sensors each have unique system level architectures which were developed by different contractors, executed using unique processes, methods, and techniques, and built with unique hardware and software sub-systems. While these individual system level architectures are well understood by the contractors who built the systems, the overall enterprise architecture containing these systems among others within the OPIR mission area is not well understood. Consequently, when the space acquisition community is studying and investigating follow-on systems to replace these legacy platforms, such as the Space Based Infrared Surveillance (SBIRS) system, this lack of detailed enterprise system of systems architecture characterization and understanding limits the government's ability to make fully educated and informed decisions about what

capabilities and functions need to be performed by the new system of systems (SoS), how legacy systems might be impacted by new systems, where commonality might exist across the systems within the enterprise, and so on. Having a comprehensive understanding of architecture considerations such as these is not only critically important for conducting trade studies, developing systems, or managing any other systems engineering activity, but is essential to understanding the impact on the overall cost, schedule, and performance levels of the SoS enterprise (e.g., reducing unnecessary redundancy and waste; reducing system level re-work; ensuring the enterprise of systems as a whole most effectively and efficiently satisfies requirements).

Model based systems engineering is a discipline that prescribes configuration controlled graphical models and views for use in managing the systems engineering activities of a system. MBSE has been widely studied and applied as a powerful systems engineering method, particularly as a tool to capture, develop, communicate, and manage system architectures. A specific MBSE architecture format or modeling language, however, has not yet emerged for DoD space applications. A relatively new systems engineering modeling language now exists—the systems modeling language (SysML). SysML has grown out of two different but related disciplines—MBSE and Software Engineering. SysML has been adopted by many organizations because it is a highly adaptable and executable modeling language, particularly concerning systems with highly complex software sub-systems. While communicating and relating system architectures across systems, organizations, and disciplines is one of the most significant challenges facing the development of a true enterprise architecture, the fact that SysML has shown itself to be flexible, adaptable, and used by many organizations makes it a strong candidate as the common standard language for developing MBSE architectures. As SysML shows high potential and promise of becoming a new standard method for conducting model based systems engineering, it is logical that it could have great potential as a common, standard, language tool for developing true mission area architectures for DoD space related mission areas, such as the OPIR mission area.

The purpose of this research is to assess model based systems engineering techniques in conjunction with methods and applications such as the enterprising system

modeling language and recommend specific applications to aid in the development of space based mission area architectures for the Department of Defense.

B. RESEARCH QUESTIONS

1. Primary research question: What methods, techniques, and processes can be employed to aid in the development of mission area architectures for Department of Defense (DoD) space systems?
2. Subsidiary research questions:
    a. In what ways or in what instances can model based systems engineering (MBSE) be used in the development of space based mission area architectures for the DoD?
    b. How can the system modeling language (SysML), based on the common Software Engineering Unified Modeling Language (UML), be applied to aid in developing mission area architectures for DoD space systems?

C. BENEFITS OF STUDY

The study being conducted during this thesis will benefit the DoD by prescribing a standardized process and framework from which model based systems engineering can be executed and enterprise architectures developed for any given mission area. Through demonstration of this technique for the Overhead Persistent Infrared (OPIR) mission area architecture, the real-world applicability and feasibility of these concepts will be explored. Organizations that could benefit from the study include the U.S. Government (USG) and any organization contracting with the USG, particularly the DoD, each military service, and the larger space acquisition community.

II. DOD ACQUISITION AND SYSTEMS ENGINEERING PROCESSES

The DoD outlines the overarching acquisition policy, procedures, and guidance to be adhered to by all military services, including the U.S. Air Force, in DoD Instruction 5000.02 *Operation of the Defense Acquisition System* (Defense Acquisition University 2013h). The System Acquisition Framework prescribed by this DoD instruction is shown in Figure 1.

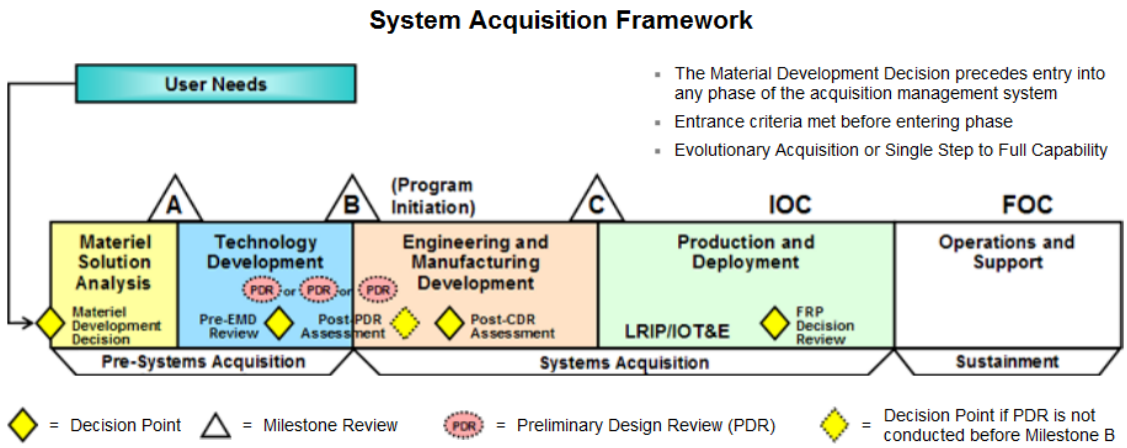

Figure 1. Department of Defense System Acquisition Framework (From Department of Defense 2008)

As shown in Figure 1, the System Acquisition Framework outlines specific phases of a major DoD acquisition program into the categories of pre-systems acquisition, systems acquisition, and sustainment. Milestones (as indicated by letters in triangles in Figure 1) separate the phases of the acquisition process, each requiring specific entrance and exit criteria for passage into the next phase. DoD Instruction 5000.02 details a wealth of specific documentation that must accompany each of the milestones, reviews, and phases outlined in the system acquisition framework shown in Figure 1. Examples of such documentation include, but are not limited to, the *initial capabilities document* (ICD), *capabilities development document* (CDD), *capabilities production document* (CPD), *systems engineering plan* (SEP), *test and evaluation master plan* (TEMP),

concept of operations (CONOPS), *system design specification* (SDS), and an *analysis of alternatives* (AoA). (Defense Acquisition University 2013a) This overarching acquisition process for the DoD, as it exists today, is a very document-focused and document-driven process in which phased documentation artifacts, including those summarized above, are generated at a specific instance in time and, in general, are used as static tools to manage the acquisition of a system.

DoD Instruction 5000.02 further defines the core disciplines necessary to implement the System Acquisition Framework (Department of Defense [DoD] 2008). Once such discipline is systems engineering, which is defined within DoD Instruction 5000.02 as "the integrating technical processes to define and balance system performance, cost, schedule, and risk within a family-of-systems and systems-of-systems context" (Defense Acquisition University 2013g, Enclosure 12). It further defines systems engineering in reference to the System Acquisition Framework by prescribing that "Systems engineering shall be embedded in program planning and be designed to support the entire acquisition life cycle" (Defense Acquisition University 2013g, Enclosure 12). While DoD Instruction 5000.02 provides one definition of systems engineering, many other definitions of systems engineering exist. The International Council on Systems Engineering (INCOSE) defines systems engineering as:

> an interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem. (INCOSE 2004)

In order to conduct the discipline of systems engineering, structured systems engineering processes must be adhered to.

One such systems engineering process has been established by the *Defense Acquisition Guidebook*, which corresponds to DoD Instruction 5000.02. The structure of this systems engineering process is maintained within the *Defense Acquisition Guidebook*, which defines the systems engineering process as "a collection of technical management processes and technical processes applied through the acquisition lifecycle" (Defense Acquisition University 2013e). DoD Instruction 5000.02 then goes on to

outline the distinct technical management processes and technical processes summarized in Table 1 (Defense Acquisition University 2013i).

Technical Management Processes	Technical Processes
Technical Planning	Stakeholder Requirements Definition
Decision Analysis	Requirements Analysis
Technical Assessment	Architecture Design
Requirements Management	Implementation
Risk Management	Integration
Configuration Management	Verification
Technical Data Management	Validation
Interface Management	Transition

Table 1. Department of Defense—Systems Engineering Processes (From Defense Acquisition University 2013i, Chapter 4)

The SE processes listed in Table 1 are conducted by the program manager and the systems engineer in an iterative, recursive, and parallel fashion throughout the acquisition life cycle. The relative degree of emphasis the program manager and systems engineer should expect to apply to each of the management and technical systems engineering processes listed in Table 1 during each phase of the System Acquisition Framework shown in Figure 1 is represented in Figure 2 (Defense Acquisition University 2013e).

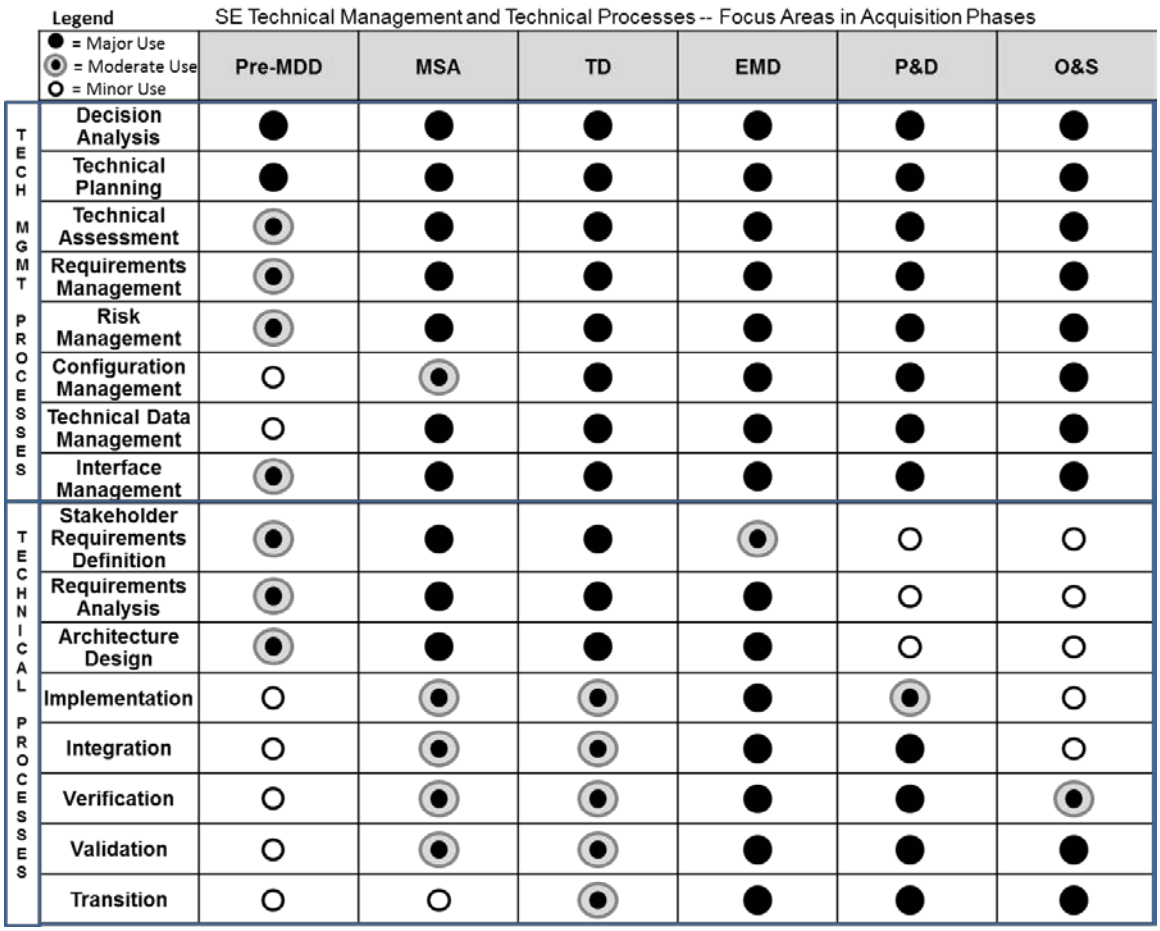

Figure 2. Notional Emphasis of Systems Engineering Processes Throughout the Defense Acquisition System Life Cycle (From Defense Acquisition University 2013e, Chapter 4)

These systems engineering technical management processes are implemented across each phase of the system acquisition framework following the "V" model structure, as shown in Figures 3 and 4.

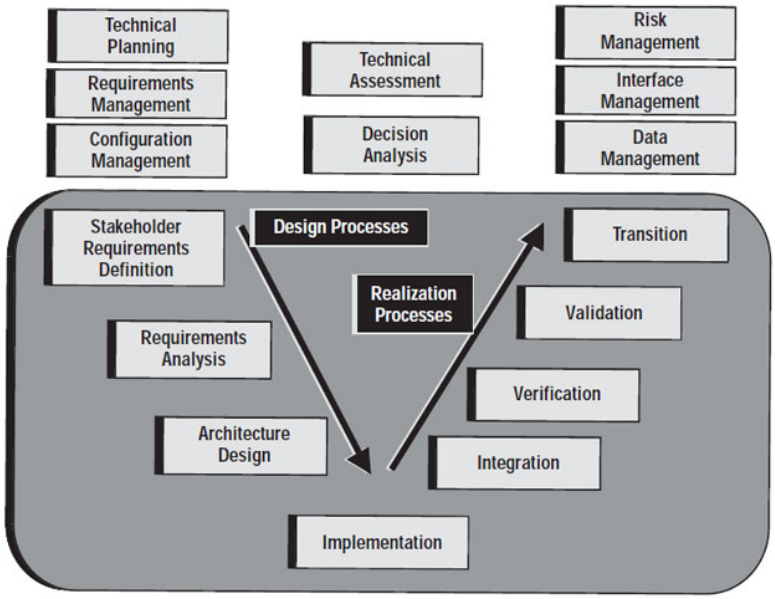

Figure 3. Department of Defense—Systems Engineering Technical Management Processes (From Defense Acquisition University 2013f)

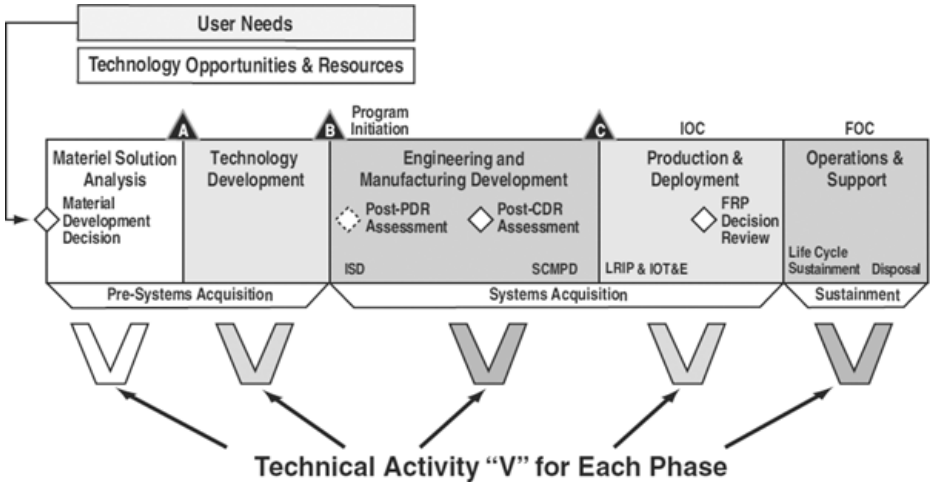

Figure 4. Department of Defense—Defense Acquisition Management System Technical "V" Activities (From Defense Acquisition University 2013f, Defense Acquisition Management System)

Each instantiation of this "V" model includes specific inputs, outputs, documentation requirements and activities for each technical management and technical process. An example of the systems engineering "V" model applied to the material solution analysis phase is shown in Figure 5. The required input and output documentation resulting from the technical systems engineering processes executed

during the material solution analysis phase of the system acquisition framework are highlighted by blue boxes.

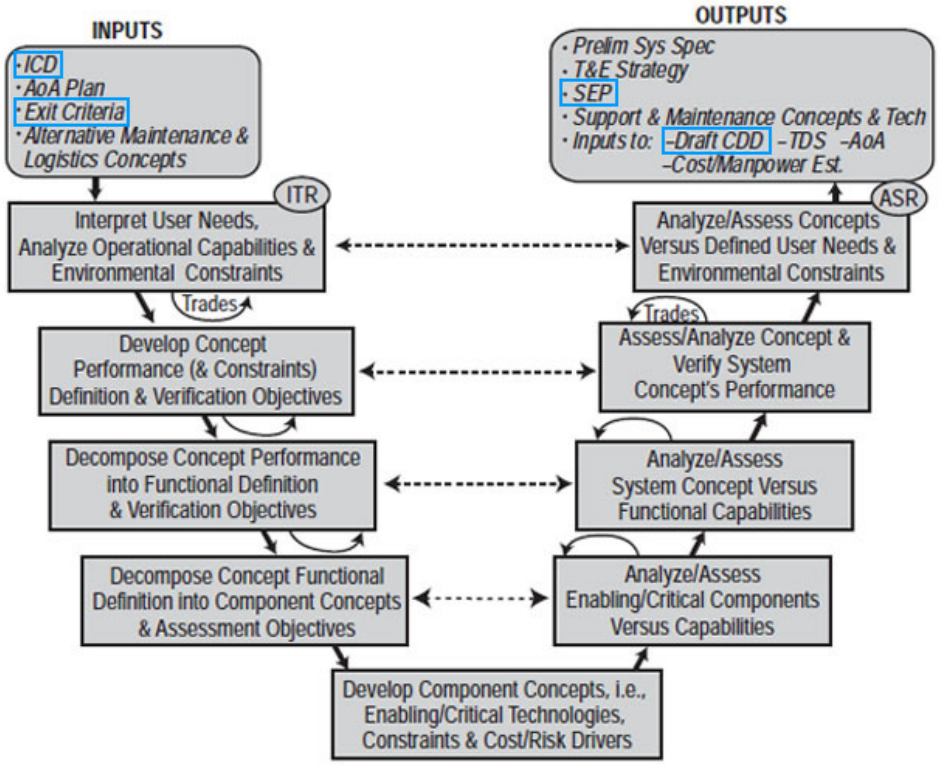

Figure 5. "V" Model Highlighting Phasing and Relationships Between Systems Engineering Activities Conducted Throughout the Materiel Solution Analysis Phase of the Defense Acquisition Framework (From Defense Acquisition University 2013j)

Of particular note from this systems engineering process prescribed by DoD Instruction 5000.02 are the requirements for documentation release at specific points in the acquisition process. Many of these documents, such as the *initial capabilities document* (ICD), *systems engineering plan* (SEP), and the *capabilities development document* (CDD) are further restricted by templates or format requirements. While the tasks, activities, and concepts of the systems engineering process described by DoD Instruction 5000.02 are flexible guidelines to meet the objectives of systems engineering, the artifacts of the process in the form of static documentation are restricted by time of release, format, and content, all of which severely limit the overall flexibility of the

process itself. Model based systems engineering, at its core, aims to shift from a document-focused systems engineering process to a repeatable and executable process that allows implementation of the tasks, activities, and concepts of the systems engineering process introduced above and defined within DoD Instruction 5000.02 while improving implementation and design flexibility through the application of dynamic products and models tailored specifically to the unique systems engineering application.

THIS PAGE INTENTIONALLY LEFT BLANK

III. MODEL BASED SYSTEMS ENGINEERING AND THE SYSTEMS MODELING LANGUAGE FOR SYSTEMS ENGINEERING AND ARCHITECTURE DEVELOPMENT

Breaking from the document-based systems engineering approach, model based systems engineering (MBSE) provides the systems engineer, architect, and designer with rigorous capabilities for conducting requirements analysis, system and sub-system design and analysis, modeling and simulation, and system verification and validation information. As is stated in *A Primer for Model Based Systems Engineering*, "in traditional systems engineering approaches, requirements reviews most often occur without adequate allocation to the physical or logical representations. Because the model-based approach addresses the allocation systematically, it leads to a better-grounded method for validating the system design" (Long and Scott 2011d, 98).

Table 2 summarizes some of the different level of features available from model-driven versus document-centered system design processes, from which we can see further benefits of a model-driven process such as MBSE over rigid document-based process such as those over-prescribed by the DoD System Acquisition Framework and DoD Instruction 5000.02.

Features	Model Driven System Design	Document Centered System Design
Information Repository	Models	Documents
Reviews (SDR, PDR, CDR)	By interrogating models (automated)	Read & interpret text then compare
Verification (FCA-Functional Configuration Audit)	Implicit, incremental, automated, built into the process	Human audit process
Communication	Reproducible and consistent	Answers may depend on readers perspective
Validation	Execute in different contexts, (e.g. customer's context, on line)	Walk-throughs, reviews of paper
Traceability –- Requirements to design to verification	Integral	Accuracy is labor intensive
Reuse	Library, "Plug and Play"	Boilerplate only
Cultural Adoption	New Paradigm	Status Quo
Infrastructure:		
Workstation & Computers	Additional computing resources	Less than model driven approach
Tools	Few Available	Extensively available
Process	Immature	Processes Exist but vary from company to company
Training	Immature	Available
Navigation	Potentially easy, since relevant data is connected	Easy to browse individual documents, but not design rationale, correlation between documents is difficult

Table 2. Comparison of Model Driven and Document Driven Approaches to System Design (From Baker, Clemente, Cohen, Permenter, Purves, and Salmon 2013)

The document-centered approach to designing and developing large complex systems brings about significant challenges in configuration management, flexibility, documentation synchronization, and enterprise collaboration. For such reasons, in *INCOSE Systems Engineering Vision 2020*, it is predicted that all systems engineering will evolve, as other engineering disciplines including mechanical, electrical, and software already have, from a document-centered to a model-driven process:

> In particular, Model Based Systems Engineering (MBSE) is expected to replace the document-centric approach that has been practiced by systems engineering in the past and to influence the future practice of systems engineering by being fully integrated into the definition of systems engineering processes. (INCOSE 2007)

A. SYSTEM MODELS

According to Long as Scott, "Models are common to human experience as aids for understanding the way the world works." (Long and Scott 2011e) In a general sense, we all use models in our daily lives to represent oftentimes more complex systems or concepts. Specific to systems engineering:

> models connect the idea behind a design solution with its implementation as a real system. These models attempt to represent the entities of the engineering problem (opportunities) and their relationships to each other and connect them to the proposed solution or existing mechanism that addresses the problem. The model used in this way is the centerpiece of MBSE. (Long and Scott 2011e)

The concept of a model can be further defined by decomposition into fundamental elements (language, structure, argumentation, and presentation) and characteristics (order, power to demonstrate and persuade, integrity and consistency, and insight). Each of these four model elements and four model characteristics, as defined by David Long and Zane Scott in *A Primer for Model-Based Systems Engineering* is further defined below: (Long and Scott 2011c, 32–33)

> **Four Elements of a Model:**
>
> **Language**—The basis for the modeling approach itself. The system definition language must be clear and unambiguous in order to depict the model accurately and understandably.
>
> **Structure**—Allows the model to capture system behavior by clearly describing the relationships of the system's entities to each other.
>
> **Argumentation**—The purpose of the model is to represent the system in such a way that the design team can demonstrate that the system accomplishes the purposes for which it is designed. Therefore the model must be capable of making the critical "argument" that the system fulfills the stakeholders' requirements.
>
> **Presentation**—Not only must the system be capable of making that argument, but it must include some mechanism of showing or "presenting" the argument in a way that can be seen and understood. (Long and Scott, 2011c)
>
> Long and Scott also elaborate on the four characteristics of a system model:

Four Characteristics of a System Model:

Order—Allows the design team to attach the problem in a coherent and consistent manner leading to a viable solution. The model provides the order that becomes the framework for this effort.

Power to Demonstrate and Persuade—By representing the relevant behaviors in proper relationship to the system entities, the model allows the designer to see and demonstrate the necessary system behavior. This becomes persuasive in making the case that a given solution answers the needs that drive the design of the system.

Integrity and Consistency—Ambiguity and inconsistency in the system design lead to design flaws which, in turn, harm the credibility of the argument that the system design meets the needs it was designed to meet. The model must, therefore, provide the integrity and consistency that lead to a sound solution.

Insight—The model provides insight into the system problem facing the design team as well as the potential design solutions. By the model's representation of the system behaviors and relationships, the design team is able to gain insight into the comparative advantages of different approaches to solving the design problem at hand. (Long and Scott 2011b, 32–33)

As Long and Scott highlight through their definitions of model elements and characteristics, models are customizable and adaptable tools which can be used by a systems engineer to design and gain unique insight into a system. There are many different modeling techniques and languages in use today that were developed to fit these definitions. For instance, examples of modeling languages for systems engineering applications include the system definition language (SDL), which is used for the structured analysis approach (realized by a modeling tool such as Vitech CORE) and the systems modeling language (SysML), which is elaborated on later as a focus of this report.

B. MODEL BASED SYSTEMS ENGINEERING

Model based systems engineering describes a set of interrelated models and views used to characterize and analyze a system design throughout its lifecycle. In *Systems Engineering Vision 2020*, the International Council on Systems Engineering (INCOSE) defines model based systems engineering as the "formalized application of modeling to

support system requirements, design, analysis, verification and validation, beginning in the conceptual design phase and continuing throughout development and later life cycle phases" (INCOSE 2007, 15).

The principles of model based systems engineering provide the framework for organizations to select a set of interrelated models to help characterize and analyze a system and document the design, acquisition and sustainment process. Using MBSE, organizations can select a set of models and views catered to meet their specific needs and, through this application, realize significant improvements to their processes and ultimately the products they produce. The selection and use of common models throughout and across organizations will help to improve communications between stakeholders, managers, and developers by providing a common ground for discussion. By standardizing these models, managers and developers can not only communicate more effectively within their program but also between multiple programs across their organization. This open communication between programs can help to facilitate the open exchange of ideas and lessons learned from one program to another, combating the "stove-piped" structure often seen between programs within a large organization or product center.

In their book, *A Practical Guide to SysML—The Systems Modeling Language* (2012a), Friedenthal, Moore, and Steiner advocate the benefits of MBSE by highlighting that it:

> …provides an opportunity to address many of the limitations of the document-based approach by providing a more rigorous means for capturing and integrating system requirements, design, analysis, and verification information, and facilitating the maintenance, assessment, and communication of this information across the system's life cycle. (20)

Friedenthal, Moore, and Steiner (2012a) further list the potential benefits of MBSE, which include:

> Enhanced communications
>
> - Shared understanding of the system across the development team and other stakeholders
> - Ability to integrate views of the system from multiple perspectives

Reduced development risk

- Ongoing requirements validation and design verification
- More accurate cost estimates to develop the system

Improved quality

- More complete, unambiguous, and verifiable requirements
- More rigorous traceability between requirements, design, analysis, and testing
- Enhanced design integrity

Increased productivity

- Faster impact analysis of requirements and design changes
- More effective exploration of trade-space
- Reuse of existing models to support design evolution
- Reduced errors and time during integration and testing
- Automated document generation

Leveraging the models across life cycle

- Support operator training on the use of the system
- Support diagnostics and maintenance of the system

Enhanced knowledge transfer

- Capture of existing and legacy designs
- Efficient access and modification of the information. (Friedenthal, Moore and Steiner 2012a, 20)

In addition to facilitating better communication and understanding throughout an organization, MBSE also improves the quality of the information presented by these models and facilitates reuse of that information. Model based systems engineering describes the use of a common database to integrate and relate the information presented by multiple models and views. There are significant benefits to employing a common integrated database to a modeling approach. At the 22nd Annual INCOSE International Symposium, representatives of the Boeing Company summarized these "benefits of MBSE in an integrated environment" (Gau Pagnanelli, Sheeley and Carson 2012), listed below:

- Single data environment ensures completeness & consistency of design data
- Rich database permits multi-user input and immediate synchronization, improving efficiency and productivity
- Use of a single data environment results in data availability throughout program life-cycles
- Traceability through model elements enables efficient change/impact analysis enabling a more adaptable system

- Robust query engine allows rapid assessment of the integrated database, finding anomalies early, preventing rework. (Gau Pagnanelli, Sheeley and Carson 2012)

By using an integrated database environment to capture the model data, valuable real-time relationships between the information, models, and the decisions they support can be realized, significantly improving the value of each model and the maintenance of the supporting data. For instance, requirements can be presented in a hierarchy showing the parent-child relationships among and between them, and source and issue linkages can be maintained as the requirements evolve. Using the common database concept, this requirements model can then be integrated with other models, such as the design architecture for a system and the risk analysis tracking tools for the associated program. The allocation of these requirements to specific architecture components and/or functions can then be shown along with the traceability of these requirements to the mission level requirements (Baker and Christian 2013). More and more complex relationships can be defined between the many models used to characterize a system and relate design issues to management initiatives associated with risk, configuration control and interface management. The common database helps with the maintenance of these interrelated models and views by allowing managers and engineers to make a single change to the database and observe the change uniformly across all applicable models and views.

C. SYSTEM ARCHITECTURE DESIGN AND DEVELOPMENT

A critical part of the systems engineering process, system architecture design is the primary tool used by systems engineers and system architects for much of the up-front system design and definition work throughout the pre-systems acquisition phase of the system acquisition framework. As is the case with the systems engineering discipline, there are many diverse definitions of system architecture. In the *INCOSE Systems Engineering Handbook v. 3.2.2*, system architecture is defined as "the arrangement of elements and subsystems and their functional allocation to meet system requirements" (INCOSE 2011, 96). The *INCOSE Systems Engineering Handbook* (SEH) further expands this definition by stating that "system… architectures depict the summation of a system's entities and capabilities at levels of abstraction that support all

stages of deployment, operations, and support" (INCOSE 2011, 98). The *DoD Architecture Framework* (DoDAF) defines system architecture as "the structure of components, their relationships, and the principles and guidelines governing their design and evolution over time" (DoD 2009, 249). In other words, system architecture is all-encompassing of a system's design and description and is an evolutionary process—two tenets that correlate strongly to the principles of MBSE which outlines evolving products used to design and capture the entirety of a system design.

As previously discussed, system architecture design and development is an iterative and recursive process. DoD Instruction 5000.02 identifies the key activities of the architecture design process as:

- Analysis and synthesis of the physical architecture and the appropriate allocation
- Analysis of the constraint requirements
- Identify and define physical interfaces and system elements
- Identify and define critical attributes of the physical system elements, including design budgets (e.g., weight, reliability) and open system principles. (Defense Acquisition Guidebook 2013b, 4.3.12)

D. WHY FOCUS ON SYSTEM ARCHITECTURE AND TRADE STUDIES?

Joseph Elm and Dennis Goldenson of Carnegie Mellon conducted a study assessing the business case for systems engineering. (Elm and Goldenson 2012) Figures 6–8 summarize the results of their study in representing the correlation between the quality of the systems engineering processes and techniques that were applied during development and their impact to the overall project performance. The study defines the term systems engineering capability (SEC) to measure the rigor of SE activities applied to a project. In addition to assessing the total SE activities (SEC-Total) applied to a project, the study decomposed SEC-Total into 11 measures of SE capability, including product architecture (SEC-ARCH) and trade studies (SEC-TRD) (Elm and Goldenson 2012, 12)

In their study, Elm and Goldenson apply the mathematical principle of Goodman and Kruskal's *Gamma*, a measure of the relative correlation or strength between two

variables. *Gamma* values can range from "-1" which indicates a very strong opposing relationship to "+1" which indicates a very strong supporting relationship. A *Gamma* value of zero indicates that there is no relationship between the two variables in question. As can be seen in Figure 6, Elm and Goldenson assessed a *Gamma* value of +0.41 for Product Architecture, indicating that there is a significant positive correlation between Product Architecture efforts and the overall performance of a project. Furthermore, the figure shows there to be nearly as strong a correlation between the quality of trade studies conducted and the project performance, a supporting fact that will later provide additional justification to SySML techniques for MBSE and system architecture design and development.

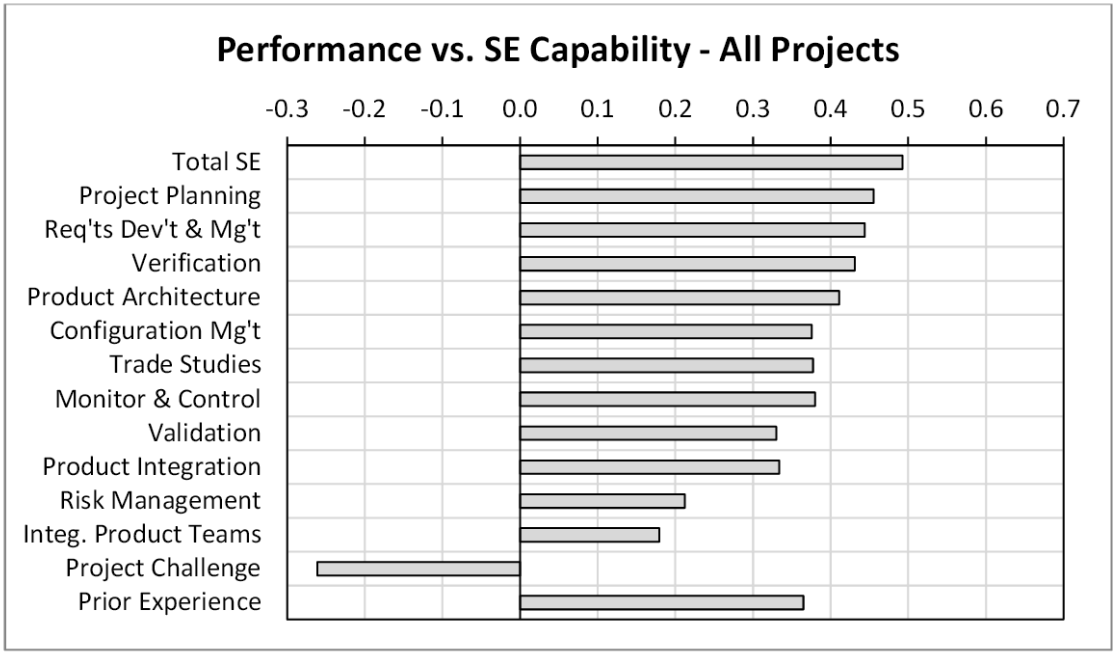

Figure 6. Strength of Correlation Between Various Systems Engineering Capabilities/Drivers and Overall Project Performance (From Elm and Goldenson 2012, Executive Summary)

Elm and Goldenson's study results further elaborate on this correlation between systems engineering activities such as architecture and trade studies and overall project performance by showing how project performance increases as the systems engineering activity level of effort increases from "lower" to "middle" to "higher." Figure 7

highlights this strong supporting relationship between architecture development and the project performance. As can be observed in Figure 7, the percentage of projects delivering higher overall performance (y-axis) increases from 16 percent to 31 percent to 49 percent as the level of product architecture efforts, or SEC-ARCH (x-axis), increases from low to middle to high, respectively.

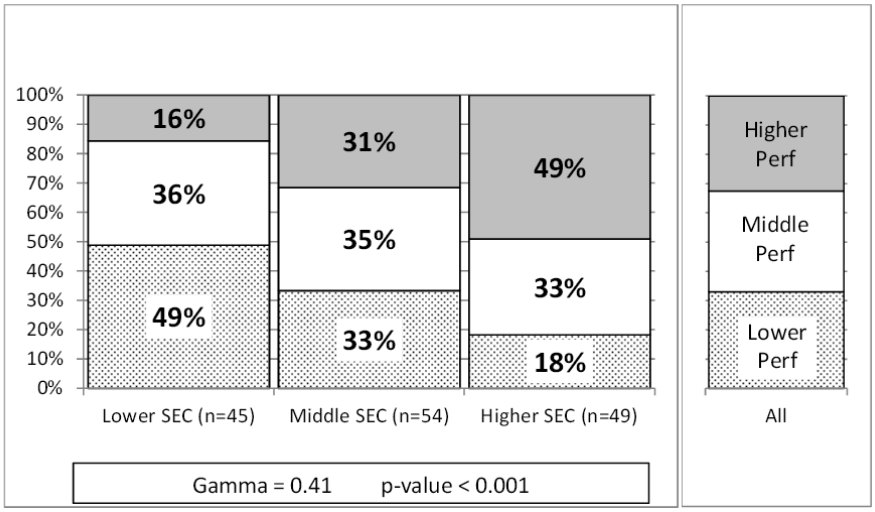

Figure 7. Mosaic Chart Comparing Various Level of SEC-ARCH to Overall Project Performance (From Elm and Goldenson 2012, 35)

Figure 8 shows a similarly strong supporting relationship between trade studies and project performance, identifying an increase in overall project performance from 13 percent to 33 percent to 52 percent as the SEC-TRD increases from low to middle to high.

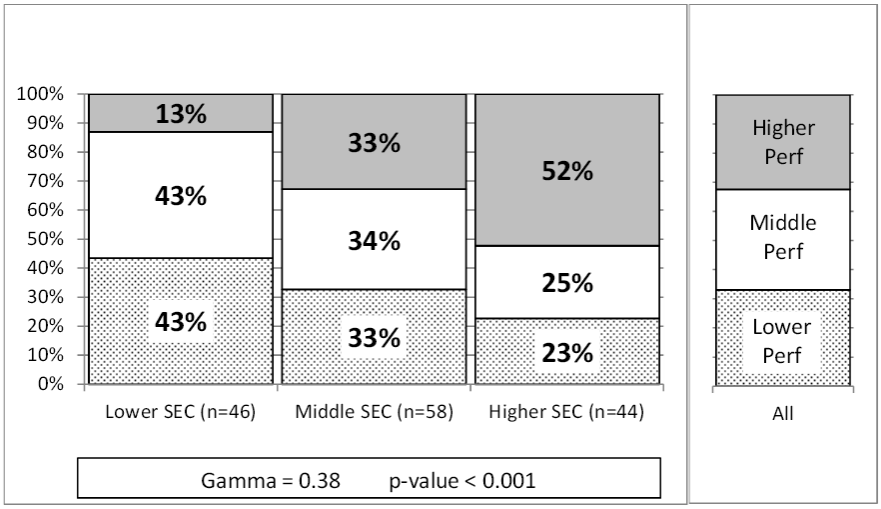

Figure 8. Mosaic Chart Comparing Various Level of SEC-TRD to Overall Project Performance (From Elm and Goldenson 2012, 38)

E. MODELING AND SIMULATION

Model based systems engineering is a powerful tool for system engineers, designers and architects because it provides strong structural support to simulation. Like models, simulations are systems engineering tools used by multiple functional disciplines throughout all lifecycles of a system. DoD Instruction 5000.02 defines modeling as "an essential [tool] to aid the understanding of complex systems and system interdependencies, and to communicate among team members and stakeholders." It relates simulation to modeling by stating, "simulation provides a means to explore concepts, system characteristics, and alternatives; open up the trade space; facilitate informed decisions and assess overall system performance" (Defense Acquisition University 2013c, 4.3.19.1). Elaboration on this definition is provided in DoD Instruction 5000.02 in the summary of the benefits of modeling and simulation listed below:

> Provides insight into program cost, schedule, performance, and supportability risk
>
> Promotes understanding of capabilities and the requirements set
>
> Provides data to inform program and technical decisions
>
> Promotes efficient communication and shared understanding among stakeholders about relationships between system requirements and the

system being developed, through precise engineering artifacts and traceability of designs to requirements

Enables better analysis and understanding of system designs (including system elements and enabling system elements), therefore providing a greater understanding of the reasons for defects and failures at all levels

Promotes greater efficiencies in design and manufacturing by reducing the time and cost of iterative build/test/fix cycles

Provides timely understanding of program impacts of proposed changes. (Defense Acquisition University 2013c, 4.3.19.1)

The activities and benefits of modeling and simulation to each phase of the System Acquisition Framework are summarized in Figures 9 and 10.

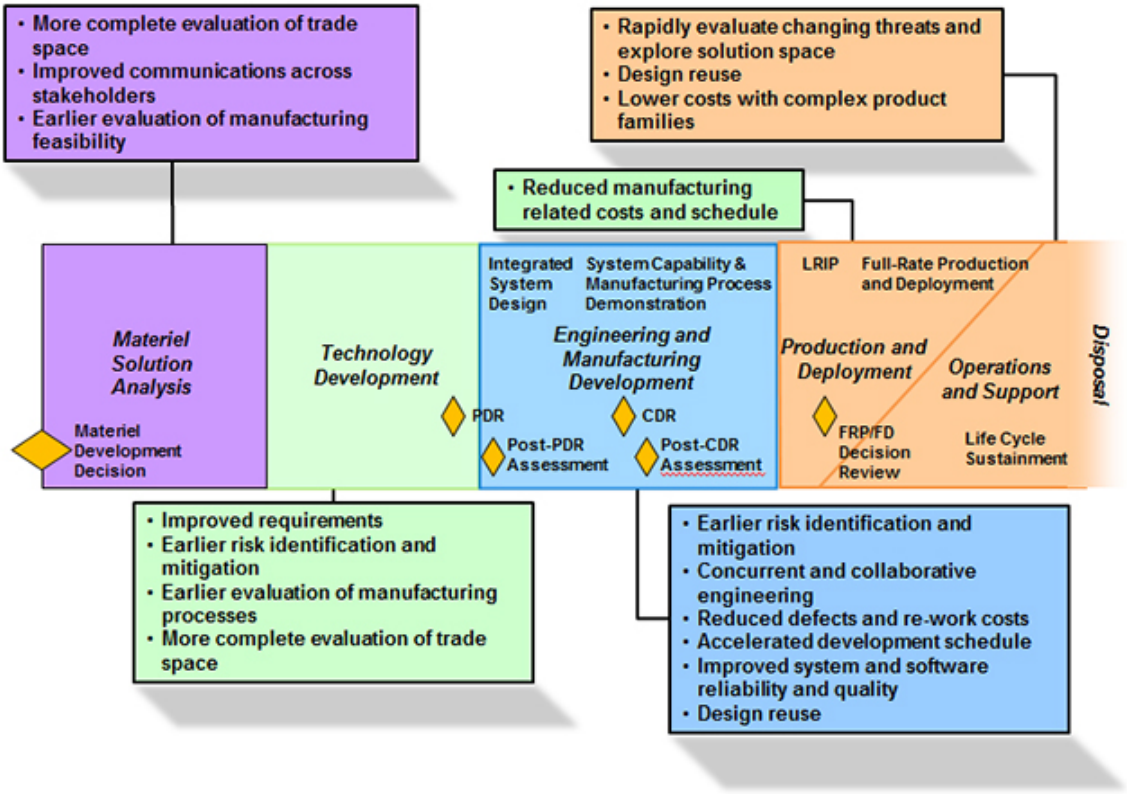

Figure 9. Benefits of Using Modeling and Simulation Throughout the Acquisition Life Cycle (From Defense Acquisition University 2013c, 4.3.19.1)

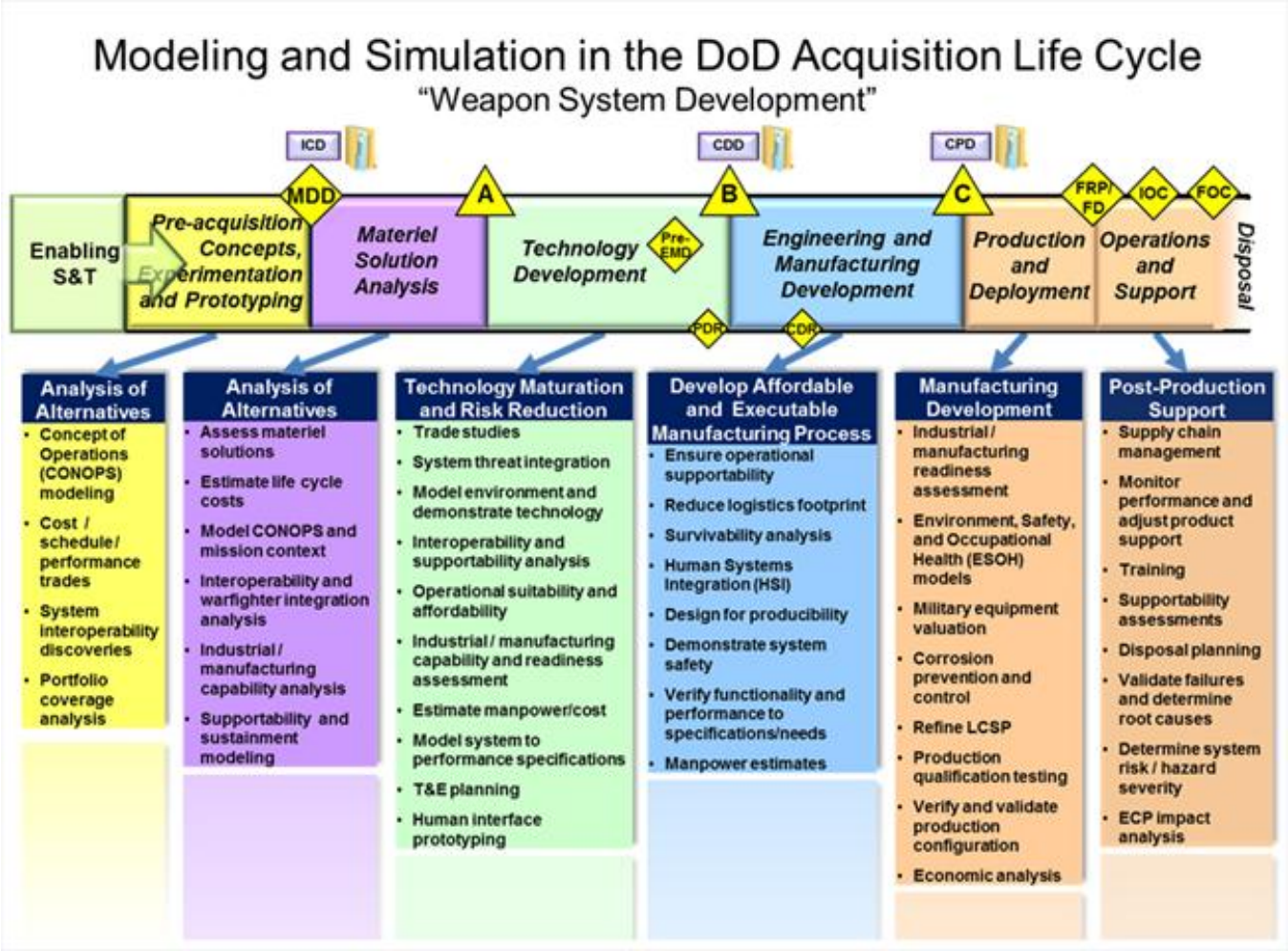

Figure 10. Various Applications of Modeling and Simulation Across the DoD Acquisition Framework (From Defense Acquisition University 2013c, 4.3.19.1)

As previously discussed, a simulation activity with particularly strong correlation to strong project performance is the analysis of alternatives, otherwise known as trade studies. While architecture design and development and trade study activities are both critical to the success of a project, they also happen to be highly correlated and interdependent activities throughout the pre-system acquisition phase of the acquisition lifecycle. It is the combination of the benefits realized as a result of rigorous architecture design and development and trade study activities that drive the MBSE approach, which is particularly well suited to strongly support both of these activities. Furthermore, as it will be shown, it is the opinion of the author of this report that the systems modeling language (SysML) is a particularly well suited approach to MBSE, which in turn optimizes both architecture design and development and trade study activities to realize the maximum potential of the MBSE approach to systems engineering.

F. ANALYSIS OF ALTERNATIVES

Section 3.3 of the *Defense Acquisition Guidebook* defines an analysis of alternatives (AoA) as "an analytical comparison of the operational effectiveness, suitability, and life-cycle cost (or total ownership cost, if applicable) of alternatives that satisfy established capability needs" (Defense Acquisition University 2013a, 3.3). In more general systems engineering terms, the AoA activity outlined by DoD Instruction 5000.02 is a trade study.

As part of an AoA or trade study, a team of engineers and analysts must conduct a comparison of competing system concepts and solutions which satisfy a set of requirements, and this must be done by assessing a broad range of system measures. These measures are analyzed across the system architecture hierarchy for each system, compared against component level or activity level measures of effectiveness (MOEs), and ultimately compiled into a top-level effectiveness MOE for each system. MBSE techniques and established system architectures are critical components in support of determining this effectiveness MOE for each system, as will be shown in the case study within this report. As highlighted by the DoD 5001, "The modeling effort should be

focused on the computation of the specific measures of effectiveness established for the purpose of the particular study" (Defense Acquisition University 2013a, 3.3).

In addition to computing the overall effectiveness of each system concept, an AoA or trade study, must estimate the total life cycle cost for each system. Once the overall effectiveness or performance and estimated life cycle cost of each system is derived, cost-effectiveness comparisons can be developed and presented as powerful tools to decision makers to ultimately select one system concept to implement from among all concepts considered within a trade study. An example of a cost-effectiveness comparison is shown in Figure 11.

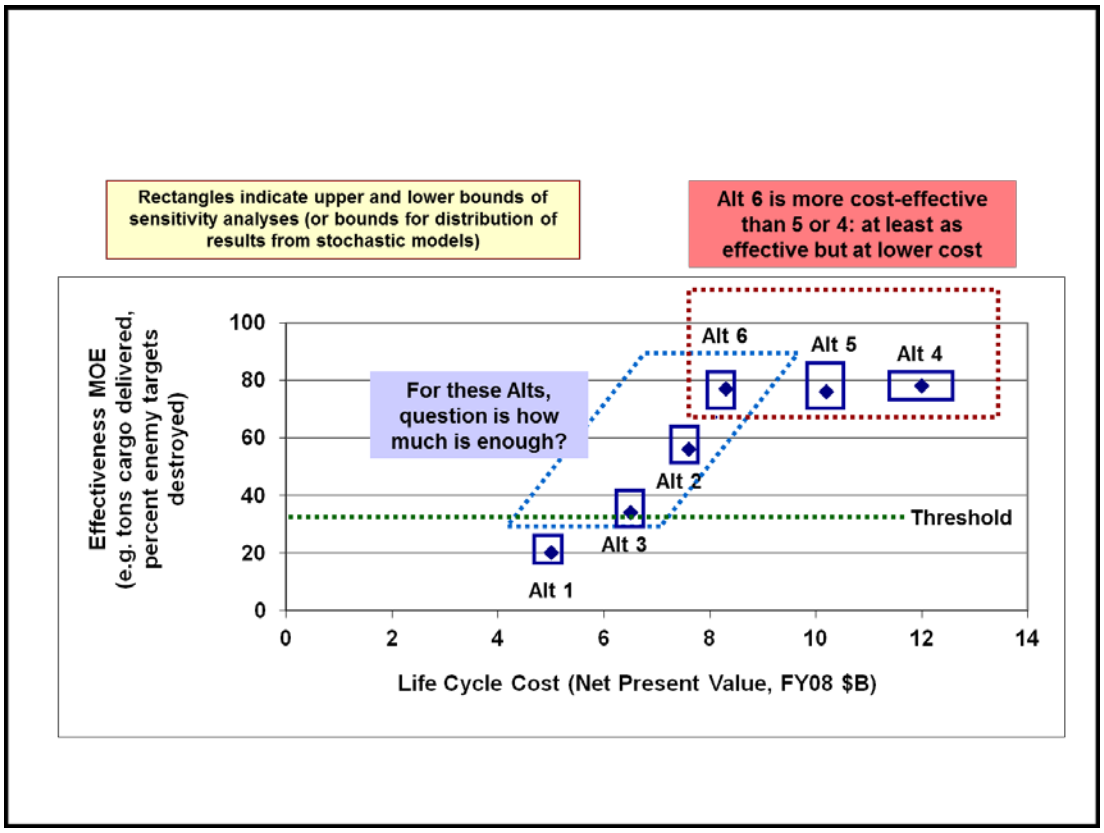

Figure 11. Cost-Effectiveness Comparison—Sample Scatter Plot of Effectiveness vs. Cost (From Defense Acquisition University 2013a, Chapter 3.3)

The cost-effectiveness comparison plot in Figure 11 is an example of how the overall system effectiveness, plotted on the y-axis, can be compared to the life cycle cost,

plotted on the x-axis, for each system alternative considered within a trade study. As shown in Figure 11, "Alt 4" and "Alt 5" are dominated by "Alt 6," which is expected to achieve the same level of effectiveness as the other dominated alternatives but at a lower life cycle cost. Taking into additional consideration any technical or schedule risk and other programmatic aspects, a decision maker could use a cost-effectiveness comparison plot like this to inform the selection of the system alternative to continue through the acquisition life cycle.

G. MBSE ARCHITECTURE TOOLS AND TECHNIQUES

There are many tools and techniques that support MBSE and are in use by systems engineering and systems architects around the world. While it is beyond the scope of this thesis to analyze and assess each of these tools and techniques, the most commonly used MBSE affiliated processes are introduced below.

1. Department of Defense Architecture Framework (DoDAF)

A very visible framework is the Department of Defense Architecture Framework, i.e., the DoDAF. As is stated in *DoDAF v. 2.0*:

> The Department of Defense Architecture Framework (DoDAF), Version 2.0 is the overarching, comprehensive framework and conceptual model enabling the development of architectures to facilitate the ability of Department of Defense (DoD) managers at all levels to make key decisions more effectively through organized information sharing across the Department, Joint Capability Areas (JCAs), Mission, Component, and Program boundaries. (Department of Defense, DoDAF v. 2.0 2009, 2)

DoDAF defines a way of representing an enterprise architecture that enables stakeholders to focus on specific interests, while retaining sight of the big picture:

> To assist decision-makers, DoDAF provides the means of abstracting essential information from the underlying complexity and presenting it in a way that maintains coherence and consistency. One of the principal objectives is to present this information in a way that is understandable to the many stakeholder communities involved in developing, delivering, and sustaining capabilities in support of the stakeholder's mission. (Defense Acquisition University 2013d, Chapter 7.2.5)

The DoDAF describes specific viewpoints from which each stakeholder can view the overarching model. Each of these viewpoints is designed to organize information from the architecture model and present it using models (e.g., graphs, tables, figures.) catered to a specific audience who can then use the model insights for system design and for decision making purposes. The viewpoints outlined by DoDAF 2.0 are summarized in the next section and shown in Figure 12.

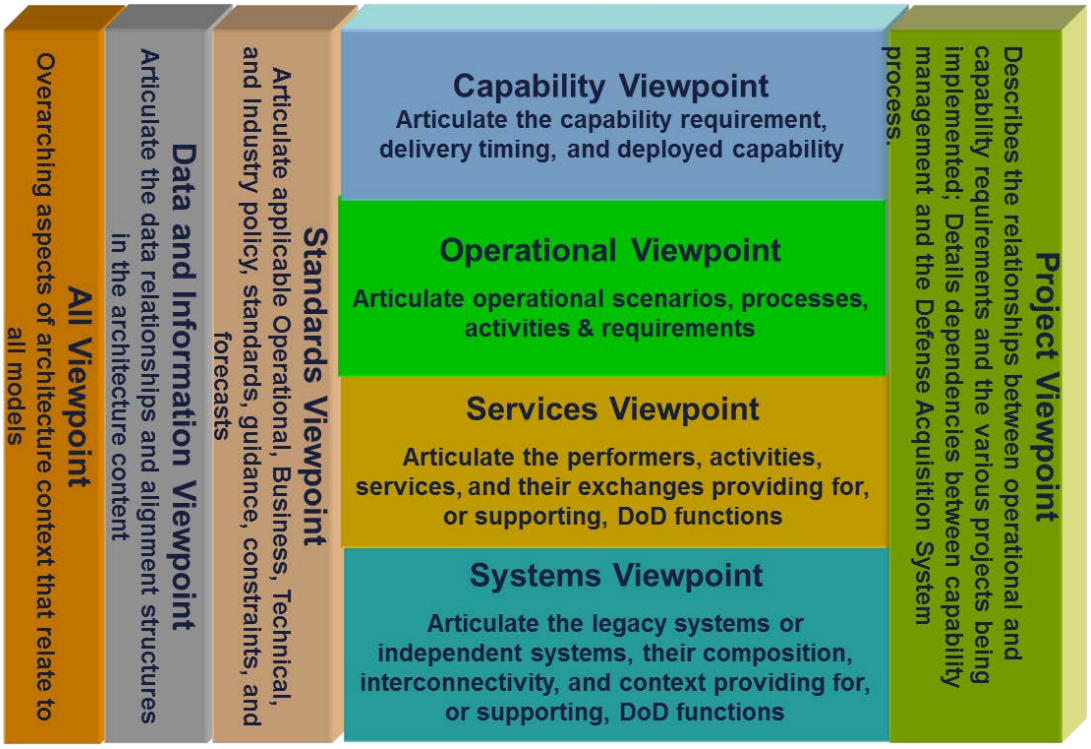

Figure 12. DoD Architecture Framework v. 2.0—Viewpoint (From Department of Defense 2009, 140)

The DoDAF defines each of these viewpoints as summarized below:

All Viewpoint: describes the overarching aspects of architecture context that relate to all viewpoints.

Capability Viewpoint: articulates the capability requirements, the delivery timing, and the deployed capability.

Data and Information Viewpoint: articulates the data relationships and alignment structures in the architecture content for the capability and operational requirements, system engineering processes, and systems and services.

> **Operational Viewpoint**: includes the operational scenarios, activities, and requirements that support capabilities.
>
> **Project Viewpoint**: describes the relationships between operational and capability requirements and the various projects being implemented. The Project Viewpoint also details dependencies among capability and operational requirements, system engineering processes, systems design, and services design within the Defense Acquisition System process. An example is the V-charts in Chapter 4 of the Defense Acquisition Guide.
>
> **Services Viewpoint:** the design for solutions articulating the Performers, Activities, Services, and their Exchanges, providing for or supporting operational and capability functions.
>
> **Standards Viewpoint**: articulates the applicable operational, business, technical, and industry policies, standards, guidance, constraints, and forecasts that apply to capability and operational requirements, system engineering processes, and systems and services.
>
> **Systems Viewpoint**: the design for solutions articulating the systems, their composition, interconnectivity, and context providing for or supporting operational and capability functions. (Department of Defense 2009, 140)

DoDAF 2.0 defines specific model viewpoints, some of which are required for major system acquisitions, and provides examples of specific models that meet these viewpoint requirements. Unlike previous releases, however, the latest DoDAF 2.0 focuses on describing the meta-model which underlies its structure and does not dictate a specific model or modeling language that must be used to satisfy any particular view or viewpoint. Because of this, any MBSE tool and technique that includes a wide range of different modeling tools, types, and languages—including structured analysis and SysML—are capable of being compliant with the DoDAF 2.0 requirements.

2. Structured Analysis and Design Technique

As is described in the *Handbook of Systems Engineering and Management*, "the structured analysis approach has its roots in the structured analysis and design technique (SADT) that originated in the 1950s and encompasses structured design, structured development, the structured analysis approach of DeMarco, and structured systems analysis" (Sage and Rouse 2011, 483–484) Structured analysis and design is a process oriented approach that outlines four primary components which together describe the

functional architecture of a system. These four components and their relative interactions are represented in Figure 13, and include the process model, data model, and rule model, along with an integrated system dictionary to manage the data supporting each model category to ensure consistency. SADT also includes dynamics modeling techniques which integrate across all three of the model categories shown.

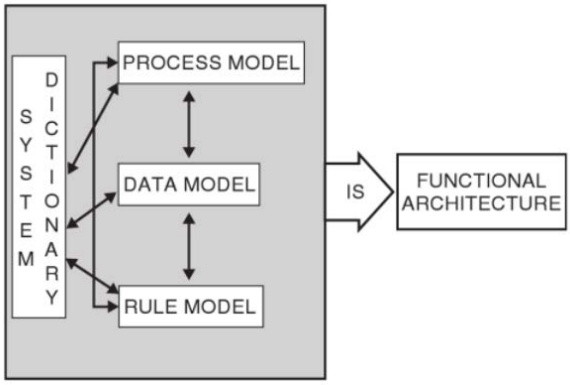

Figure 13. Components of the Structured Analysis and Design Technique (From Sage and Rouse 2011, 485)

The most commonly used model diagrams and techniques employed to the process model, data model, rule model, and dynamics model are summarized in Table 3.

SADT Model	Associated Diagrams and Techniques
Process Model	• IDEF0—Data Flow Diagrams
Data Model	• IDEF1X—Entity Relationship Diagrams
Rule Model	• Decision Trees and Tables • Structured English • Mathematical Logic
Dynamics Model	• State Transition Diagrams • Functional Flow Block Diagrams (FFBDs)

Table 3. Structured Analysis and Design Models, Diagrams, and Techniques (From Long, 2010, 7)

These models (and the integrated dictionary supporting them) support the hierarchical decomposition of a system as its architecture is further modeled and defined. For instance, the decomposition of an IDEF0 model used to capture the functional and physical architecture of a system is presented in Figures 14 and 15.

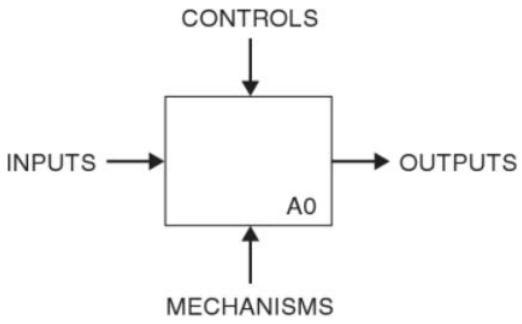

Figure 14. IDEF0 Semantic Diagram (From Sage and Rouse 2011, 486)

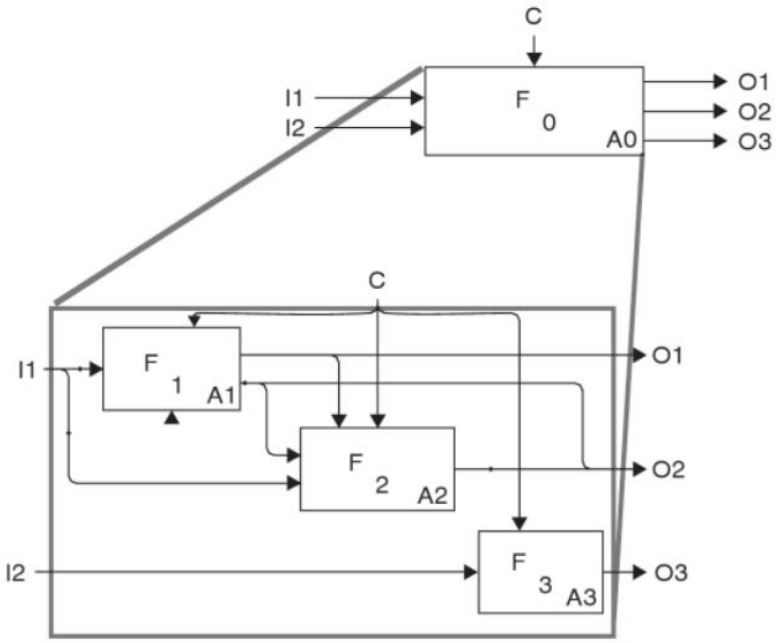

Figure 15. IDEF0 Activity Diagram—First Two Levels (From Sage and Rouse 2011, 487)

Modeling tools, such as CORE by the Vitech Corporation, provide a comprehensive integrated toolset to support MBSE techniques and the development of the models outlined by the structured analysis and design technique. Vitech's CORE tool defines a specific modeling language, called the system definition language (SDL), which "expresses and represents the model clearly, so that understanding and insight can arise" (Long and Scott 2011c, 32). A mapping of some of the components of SDL to MBSE examples is shown in Table 4.

English Equivalent	System Definition Language	MBSE Example
Noun	Entity	**Requirement:** Place Orders **Function:** Collect Images
Verb	Relationship	Requirement is the **basis of** Function
Adjective	Attribute	**Description**
Adverb	Relationship Attribute	Function consumes Resource **Amount** of Resource being consumed
N/A	Structure	**Activity Diagram** **Enhanced Functional Flow Block Diagram**

Table 4. Components of the SDL Mapped to MBSE Examples (From Long and Zane 2011a, 37)

The CORE modeling tool employs an internal SDL model taxonomy, shown in Figure 16, in order to realize the benefits of an integrated system dictionary capable of generating a wide variety of process, data, rule, or dynamics models and views, and each of these leverage the same common model database. As previously discussed, MBSE tools and techniques such as that represented by the Vitech CORE toolset can be compliant with DoDAF 2.0. In reference to the model taxonomy, shown in Figure 16, *A Primer for Model Based Systems Engineering* states:

> In the case of DoDAF, the Architecture class acts as a key element. It brings the physical natures of the operational and system sides together. Thus, in a physical sense, it is clear that a particular Architecture entity provides the context for understanding how a set of operational entities and a corresponding set of system entities relate. (Long and Zane 2011a, 37–38)

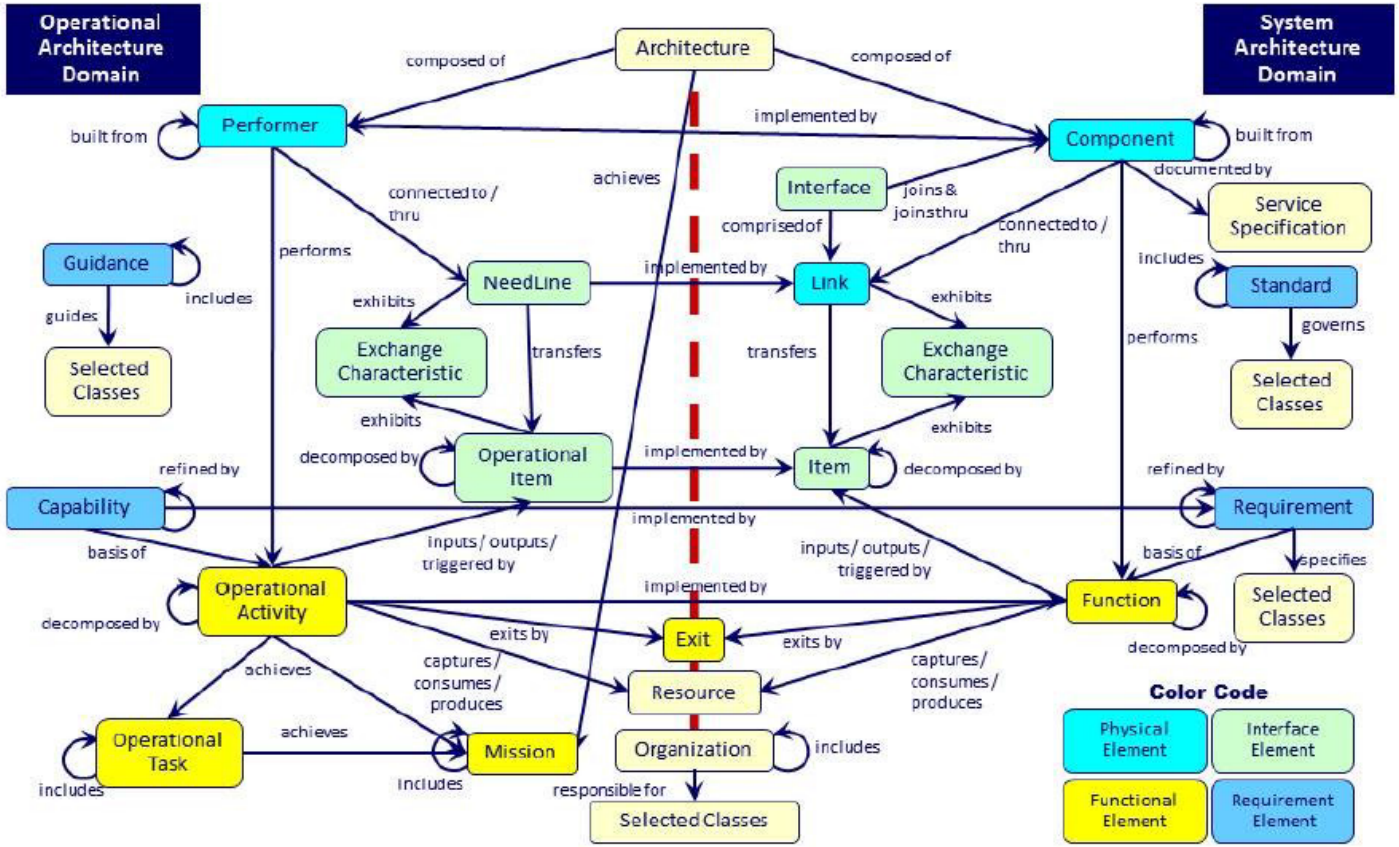

Figure 16. Relationship of the Parts of Speech From Common Language to the MBSE SDL (From Long and Zane 2011a, 38)

3. Systems Modeling Language (SySML)

SysML, like its parent language, the Unified Modeling Language (UML) for software engineering, was developed with comprehensive support to modeling and simulation to include powerful and rigorous frameworks to optimize analysis of alternatives (AoAs) and trade studies. As is stated in *OMG Systems Modeling Language V. 1.3*:

> SysML is designed to provide simple but powerful constructs for modeling a wide range of systems engineering problems. It is particularly effective in specifying requirements, structure, behavior, allocations, and constraints on system properties to support engineering analysis. (Object Management Group 2012, 25)

Like the Structured Analysis and Design Technique, SysML also complies with DoDAF requirements and is capable of generating models across all DoDAF viewpoints to realize all benefits of the enterprise architecture framework.

> Compared to the document-centric approach, which is predominately used in conceptual modeling today, SysML models offer a much more useful format in terms of reusable blocks of information. Compartmentalizing information allows it to be offered to readers in more digestible quantities; different amounts and different sections of information can be offered to readers depending on the role they play in the study. Once built in appropriate software, a SysML model also allows for more intuitive navigation through the information, again aiding the communication process. (Liston, Kabak, Dungan, Byrne, Young, and Heavey 2010, 304)

a. History of SysML

As is described in Liston et al. 2010, in January 2001, the International Council on Systems Engineering (INCOSE) Model Systems Design Workgroup made the decision to adopt and expand for systems engineering applications the Unified Modeling Language (UML), a popular tool used in the Software Engineering discipline. INCOSE began collaborating with the Object Management Group (OMG), which maintains the UML specification, and together they developed the set of requirements for SysML. In March 2003, these requirements were issued by OMG as part of the UML for systems engineering request for proposal (RFP). In response to the OMG RFP, a work group

including members from industry and tool vendors was formed in May 2003. The SysML Partners, as this group was known, initiated an open source specification project to develop the SysML standard according to the outlined requirements. In September 2007, the SysML Partners, in conjunction with the OMG, published *SysML Version 1.0*—the first official SysML standard. Following the release of the official SysML standard, another group called the SysML Revision Task Force was established to monitor the specification and recommend revisions as necessary. In December 2008, the *OMG SysML v1.1* standard was published, incorporating the first set of revisions to the standard based on inputs from across the systems engineering community. The current version, *OMG SysML v1.3*, was published by OMG in June 2012 and is the basis for all SysML discussions within this report (Liston et al. 2010, 282–283).

b. Overview of SysML

SysML reuses some components and provides extensions to UML. The Venn diagram in Figure 17 shows a representation of the interrelationship between UML and SysML (Object Management Group 2012, 7).

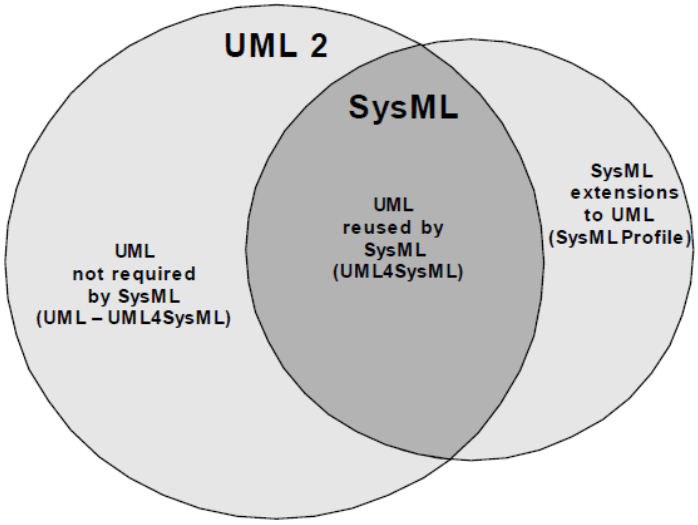

Figure 17. Overview of the SysML and UML Interrelationship (From Object Management Group, 7)

According to the OMG SysML specification Version 1.3, SysML was designed based on six fundamental principles, summarized below.

> **Requirement**s-driven: SysML was designed to satisfy the requirements of the UML for Systems Engineering RFP
>
> **UML reuse**: SysML reuses UML wherever practical to satisfy the requirements of the RFP, and when modifications are required, they are done in a manner that strives to minimize changes to the underlying language. Consequently, SysML is intended to be relatively easy to implement for vendors who support UML 2
>
> **UML extensions**: SysML extends UML as needed to satisfy the requirements of the RFP. The primary extension mechanism is the UML 2 profile mechanism
>
> **Partitioning**: The package is the basic unit of partitioning in the SysML specification. The packages partition the model elements into logical groupings that minimize circular dependencies among them
>
> **Layering**: SysML packages are specified as an extension layer to the UML metamodel
>
> **Interoperability**: SysML inherits the XMI interchange capability from UML. SysML is also intended to be supported by the ISO 10303-233 data interchange standard, otherwise known as Application Protocol 233 or AP233, to support interoperability among other engineering tools. The specific AP233 interoperability will be discussed later in this report in reference to realizing the proposed systems engineering solution at SMC and across the DoD Space Acquisition community. (Object Management Group 2012, 8)

The SysML diagram taxonomy is shown in Figure 18. The two new diagram types that have been added to SysML include the requirement diagram and the parametric diagram, as shown. Each of the diagram types is then summarized (Object Management Group 2012, 167–172).

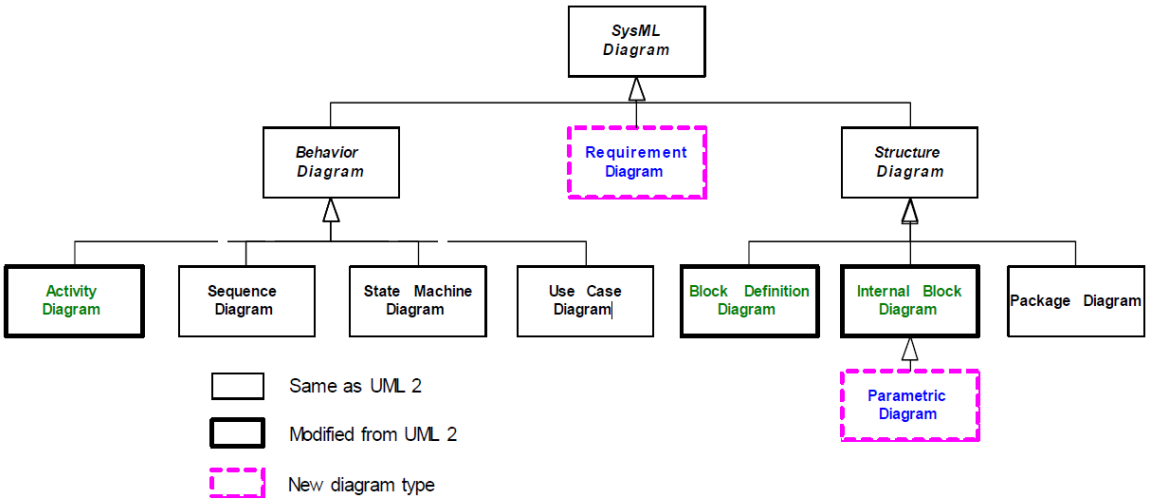

Figure 18. SysML Diagram Taxonomy (From Object Management Group 2012, 167)

The system structure is represented by block definition diagrams and internal block diagrams, which are based on the UML class diagram and UML composite structure diagram, respectively. Liston et al. (2010) summarize each of the SysML diagram types as defined below:

> The block definition diagram describes the system hierarchy and system/component classifications through the representation of structural elements called blocks. Any block that exhibits behavior must have an associated state machine diagram
>
> The internal block diagram describes the internal structure of a system in terms of its parts, ports, and connectors
>
> The parametric diagram is a restricted form of the internal block diagram and represents constraints on property values. This diagram is used to integrate behavior and structure models with engineering analysis models such as performance, reliability, and mass property models
>
> The package diagram represents the organization of a model in terms of packages that contain model elements
>
> The behavior diagrams include the use-case diagram, activity diagram, sequence diagram, and state machine diagram.
>
> The activity diagram represents the flow of data and control between activities and shows how actions transform inputs into outputs

> The sequence diagram represents the interaction between collaborating parts of a system in terms of a sequence of exchanged messages
>
> The state machine diagram describes the state transitions and actions that a system or its parts performs when triggered by events
>
> The use-case diagram provides a high-level description of the system functionality in terms of how a system is used by external entities (i.e., actors)
>
> The requirements diagram is neither structural nor behavioral. It supports requirements traceability by representing text-based requirements. It also provides a modeling construct for modeling the relationship between requirements and other model elements that satisfy or verify them. (Liston, et al. 2010, 283–285)

c. SysML Purpose and Key Features

SysML is a general-purpose graphical modeling language that provides rigorous MBSE capabilities in support of the complete system acquisition lifecycle—including analysis, specification, design, verification, and validation. It can be applied to any system, simple or complex, including but not limited to software, hardware, data processing, personnel, organizations, and procedures.

> The language is intended to help specify and architect systems and specify their components that can then be designed using other domain-specific languages such as UML for software design and VHDL and three-dimensional geometric modeling for hardware design. SysML is intended to facilitate the application of an MBSE approach to create a cohesive and consistent model of the system. (Friedenthal, Moore and Steiner, Chapter 3: Setting Started with SysML 2012b, 29)

Employing the diagrams summarized above, SysML can represent many different system aspects, including:

- Structural composition, interconnection, and classification
- Function-based, message-based, and state-based behavior
- Constraints on the physical and performance properties
- Allocations between behavior, structure, and constraints
- Requirements and their relationship to other requirements, design elements, and test cases. (Friedenthal, Moore and Steiner 2012b, 29)

Detailed examples of each of these diagrams, along with a discussion of their representation of the above system aspects, are provided in the SysML use case in the following chapter.

d. SysML Support to Modeling and Simulation

Through the use of parametric models, SysML supports a wide range of modeling and simulation and engineering analyses activities, including: trade studies, sensitivity analysis, design optimization, and analysis of performance, reliability, and physical properties of a system (Friedenthal, Moore and Steiner 2012c, 185). These parametric models are used in SysML to capture constraints on the properties of a system, which can then be computed and evaluated by an analysis tool.

> Constraints are expressed as equations whose parameters are bound to the properties of a system. Each parametric model can capture a particular engineering analysis of a design. Multiple engineering analyses can then be captured in parametric models that are related to a system design alternative, and then executed to support trade-off analysis. (Friedenthal, Moore and Steiner 2012c, 185)

As will be seen in the SysML case study in the following chapter, a special model block called the constraint block, as shown in Figure 19, is used to define equations and support the construction of parametric models.

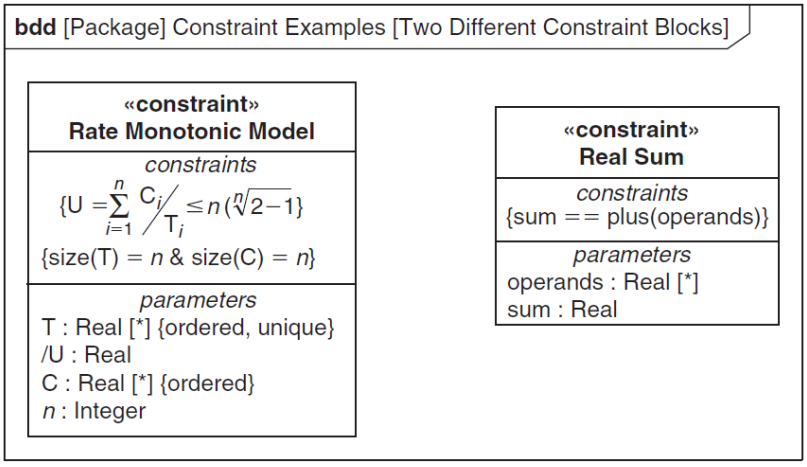

Figure 19. Two Reusable Constraint Blocks Expressed on a SysML Block Definition Diagram (From Friedenthal, Moore and Steiner 2012c, 189)

Figure 19 shows two constraint blocks, Real Sum and Rate Monotonic Model. Real Sum is a simple reusable constraint where one parameter, sum, equals the sum of a set of operands, as expressed in the constraint in the constraints compartment. (Friedenthal, Moore and Steiner 2012c, 189–190)

Constraint blocks are used in SysML to support analysis of alternatives and trade studies. Each alternative solution is defined by a set of measures of effectiveness (MOEs) that corresponding to specific evaluation criteria of the requirements levied on the system. Equations for these evaluation criteria are applied to the model and used to calculate a value for each MOE. The defined hierarchy of the SysML model is used to allocate these specific evaluation criteria values to applicable elements throughout the model. Using this model construct, the MOEs for each alternative solution can be evaluated and compared against an objective function. Results for each alternative are then compared in an analysis of alternatives to help inform decision makers. As an example, Figures 20–22 show a simple trade study evaluating two variants of a camera designed to operate in low-light conditions (Friedenthal, Moore and Steiner 2012c, 200–202).

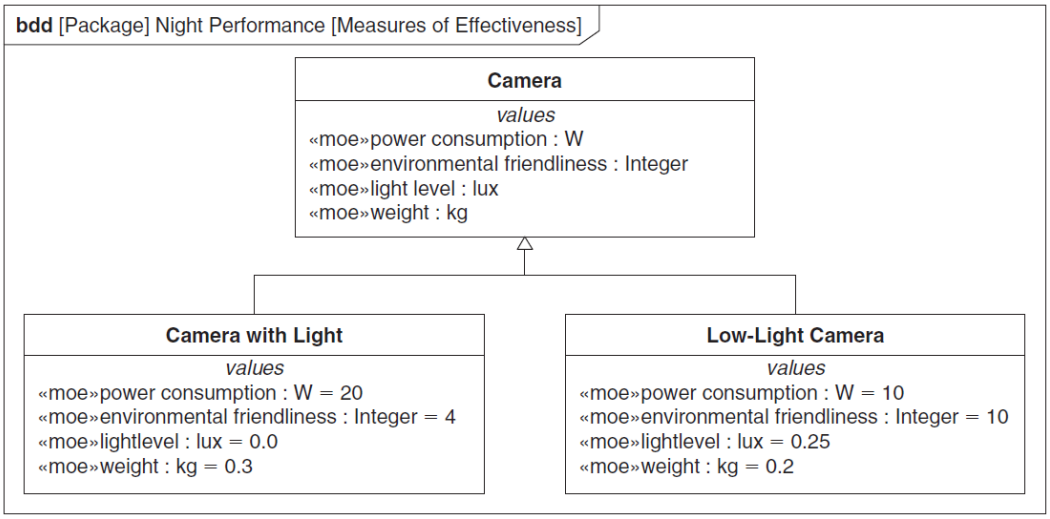

Figure 20. Two Variants of a Camera for Handling Low-Light Conditions are Defined Using a SysML Block Definition Diagram (From Friedenthal, Moore and Steiner 2012c, 201)

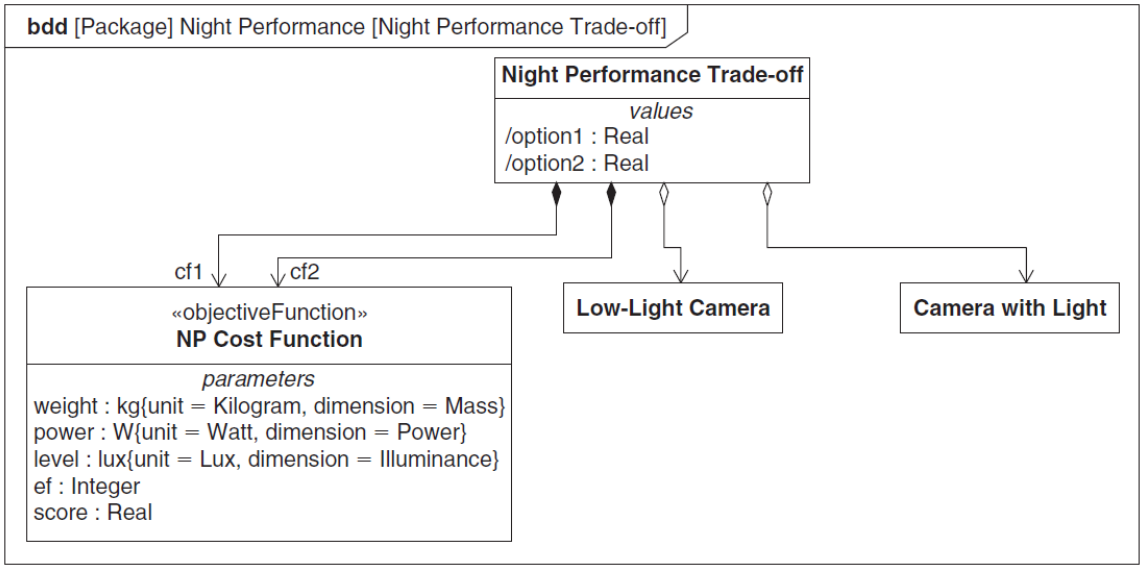

Figure 21. A SysML Block Definition Diagram Represents an Analysis Context, Laying out a Trade Study for the Two Camera Variants (From Friedenthal, Moore and Steiner 2012c, 201)

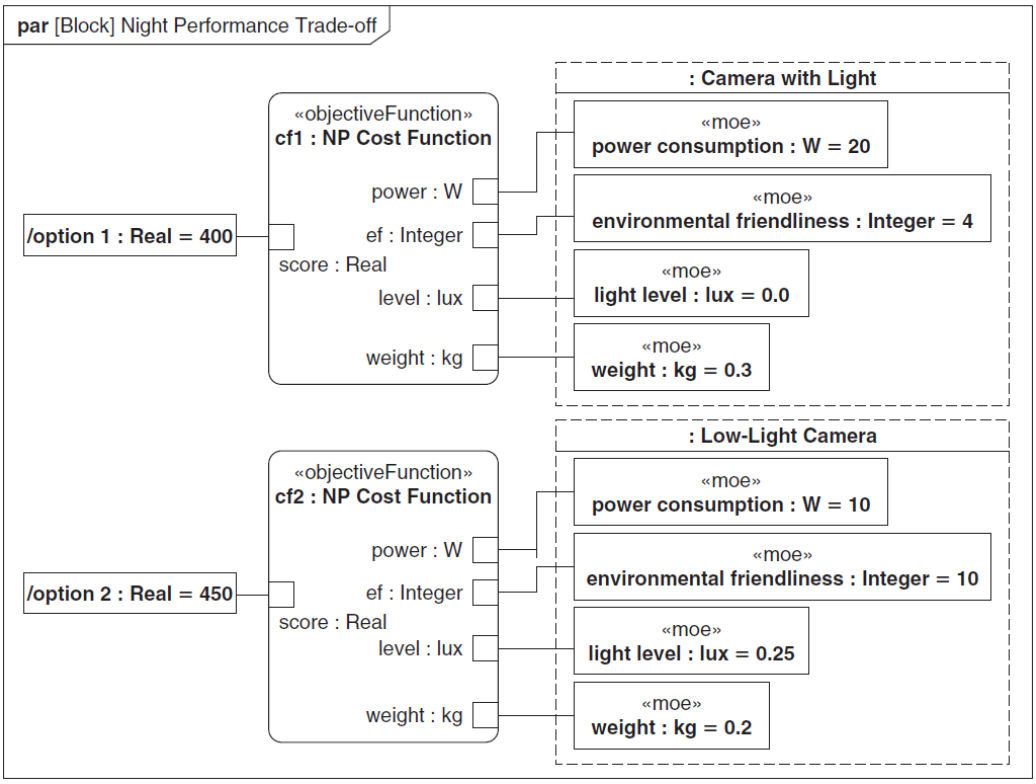

Figure 22. Trade-off Results Between the Two Low-Light Camera Variants (From Friedenthal, Moore and Steiner 2012c, 202)

In the camera example shown in Figures 20–22, the low-light camera defined as Option 2 would be the preferred solution, given its higher score of 450 over a score of 400 for Option 1, as shown in Figure 22. In order to further build out a more rigorous trade-off analysis, additional constraint blocks could be added to correspond to additional MOEs targeting other properties of the system. Another example of this approach to trade studies is shown later in the SysML case study.

e. SysML Tools

There are many commercial and open source tools available for developing SysML models. The most significant of these tools are summarized briefly below:

- Artisan Studio by Atego is a UML tool that has been developed to fully support the SysML profile ("Artisan Studio" 2013)
- Tau G2 by IBM is a standards-based, model-driven development solution for complex systems ("Rational Tau," IBM website 2013)
- Rational Rhapsody also by IBM is a UML/SysML-based model-driven development for real-time or embedded systems ("Rational Rhapsody Family," IMB website 2013)
- MagicDraw by No Magic is described as a business process, architecture, software and system modeling tool, having a specific plugin to support SysML modeling ("Magic Draw," No Magic website 2013)
- Enterprise Architect by Sparx Systems is a UML analysis and design tool with a module for developing SysML models ("Enterprise Architect," Sparx Systems website, 2013)
- CORE Spectrum and GENESYS by the Vitech Corporation provides a foundation for enhanced system modeling, system analysis, and expedited communication. Vitech has recently added some basic views in support of SysML diagrams ("SysML Modeling," Vitech website 2013)
- Modelio is an open source modeling environment with an open-source version and fully featured commercial version of an SysML plugin ("SysML Architect," Modelio Store website 2013)
- TOPCASED-SysML is a SysML editor that has been developed by the open source community ("TOPCASED-SYSML," Fusion Forge website 2013)

- Papyrus for SysML is an open source UML tool based on the Eclipse environment and includes all of the stereotypes defined in the SysML specification (Papyrus UML website 2013)

The MagicDraw tool by No Magic was selected as the SysML model for use in the Overhead Persistent Infrared (OPIR) use case detailed in the following chapter. MagicDraw was selected over the other tools primarily because of its availability—free trial versions of both MagicDraw and its SysML module were available and included access to the full features of the tool, unlike trial versions of the other systems. In addition to the commercial versions of MagicDraw and CORE, the open-source tools Modelio, TOPCASED, and Papyrus were installed and experimented with by the author of this report, but the commercial applications were much more easily adopted given their more streamlined user interfaces and more complete guidance and support documentation. MagicDraw was selected over CORE because it included much more robust SysML modeling capabilities. Additionally, MagicDraw, along with the SysML module was the tool used to develop the diagrams and example problems contained within the *OMG SysML* Version 1.3 specification, allowing for maximum commonality between the SysML guidance and the tool used to build the model for the case study.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. CASE STUDY—OVERHEAD PERSISTENT INFRARED (OPIR) MISSION AREA ARCHITECTURE

A. PURPOSE

The purpose of this case study is to demonstrate how SysML can be applied to DoD Space Systems and support architecture development and trade studies in support of decision making.

B. SCOPE

This case study is not intended to highlight every aspect of SysML, nor is it intended to reflect a complete architecture for the Overhead Persistent Infrared (OPIR) mission area. Rather, the intent of this example problem is to demonstrate how SysML diagrams can be applied and to illustrate the primary features of SysML, including the interrelationships among the different model elements and diagrams. At least one diagram of each SysML diagram type, presented in Figure 18, is included in this case study to best demonstrate these features of SysML. The structure of this case study example mimics the diagram examples provided in the *OMG SysML* Version 1.3 specification (Object Management Group 2012, Annex C) in order to maximize the correlation between this case study problem and the overarching SysML guidance documentation.

Given the assumed limitations to the OPIR architecture represented by SysML in this case study, there will be functions, elements, and objects missing from the OPIR architecture itself in each of the diagrams presented in order to maintain relative simplicity within each diagram to ensure that the SysML model elements and interrelationships can be easily identified.

C. PROBLEM SUMMARY

The DoD Overhead Persistent Infrared (OPIR) mission area is broken into four mission areas:

- **Missile Warning**
  - Provides first response reporting of launch events anywhere in world
  - Enables precise launch trajectory and impact point computation
- **Missile Defense**
  - Cues missile defense systems
  - Identifies location of launches for ground operations
- **Technical Intelligence**
  - Detects detailed missile phenomenology to characterize threats and tactics
- **Battlespace Awareness**
  - Provides insight into battlefield events and assessment of operations
  - Enables more efficient resource management

Currently, there are two major DoD space systems supporting the OPIR mission area: the Defense Support Program (DSP) and the Space Based Infrared System (SBIRS). DSP is the legacy OPIR system, first launched in 1966. SBIRS is the current DoD Acquisition Program of Record (POR) for the OPIR mission area, and is described as an integrated system of systems that includes satellites in geosynchronous orbit (GEO), sensors hosted on satellites in highly elliptical orbit (HEO), and ground-based data processing and control (LAAFB 2013). This case study focuses on defining the architecture (to include requirements, performance analyses, structure, and behavior) of the SBIRS system in support of the OPIR mission area system architecture (MASA), with a specific focus on trade study design decisions concerning the SBIRS spacecraft power subsystem.

D. SYSML DIAGRAMS

This case study will focus on demonstrating the following SysML content and diagrams, in the order listed:

- SysML diagrams used to establish the system context, system boundaries, and top-level use cases
- SysML diagrams used to analyze top-level system behavior using sequence diagrams and state machine diagrams
- SysML diagrams for capturing and deriving requirements

- SysML diagrams and techniques to depict system structure using block hierarchy and part relationships
- SysML diagrams and techniques to illustrate the relationship of system parameters, performance constraints, analyses, and timing diagrams
- SysML diagrams and techniques depicting the interfaces and flows in a structural context
- SysML diagrams and techniques capturing behavior modeling and functional flow allocation

1. Internal Block Diagram—System Context

The SysML Internal Block Diagram shown in Figure 23 shows an example of a top level system enterprise and the external systems and actors which relate and interact with the system. Such a diagram would likely be user-generated and satisfy the DoDAF requirement for an Operational View 1 (OV-1). The various elements of the context diagram are represented by SysML blocks, and the type of block is identified at the top of the block. For example, block types in the context diagram in Figure 23 include <<system>> and <<external>> stereotypes. These "stereotypes" are defined by the system modeler and, while they are not structured terms of the SysML language, they are useful classifications for the modeler and user to identify the system of interest relative to its environment. At this point in the model development, the associations between the blocks (shown as connecting lines) represent abstract conceptual relationships between the entities. These relationships will be defined in greater detail by subsequent SysML diagrams.

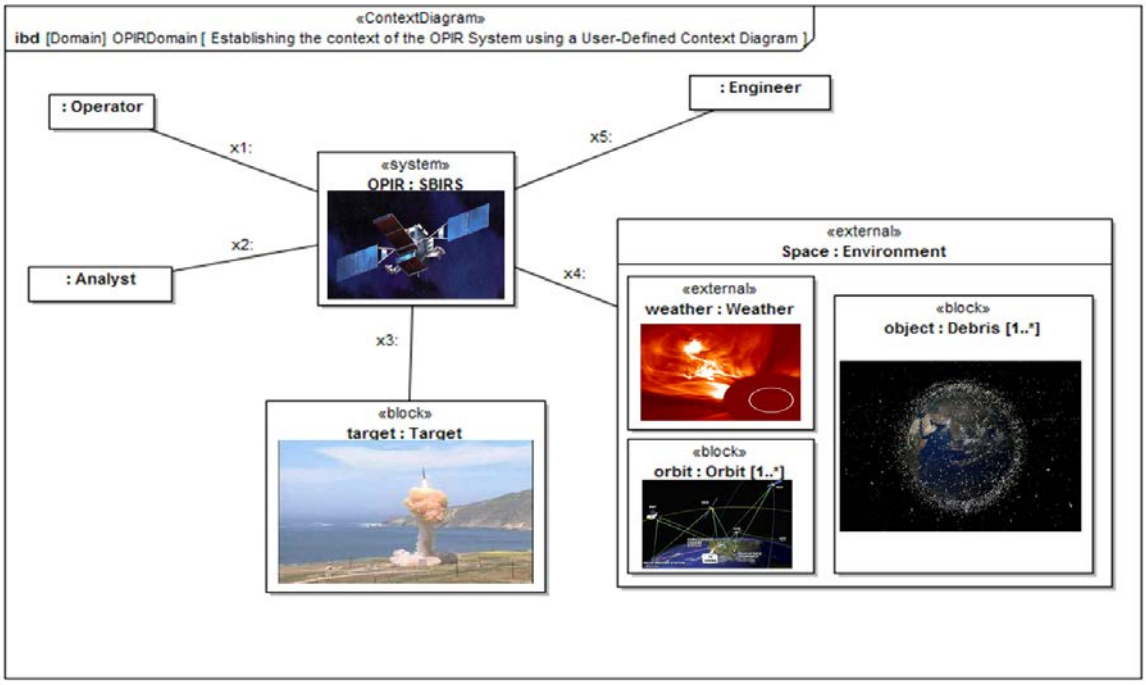

Figure 23. SysML Internal Block Diagram Establishing the Context of the OPIR System Using a User-Defined Context Diagram

2. Use Case Diagram—Top Level

A SysML use case diagram for the SBIRS system is shown in Figure 24. The SBIRS system is shown as the subject of the use case diagram and the actors (including the operator, Air Force, engineering, prime contractor, and Department of Defense) are shown interacting with component or child use cases (i.e., operate SBIRS, develop the system, operationalize the system, and maintain the system).

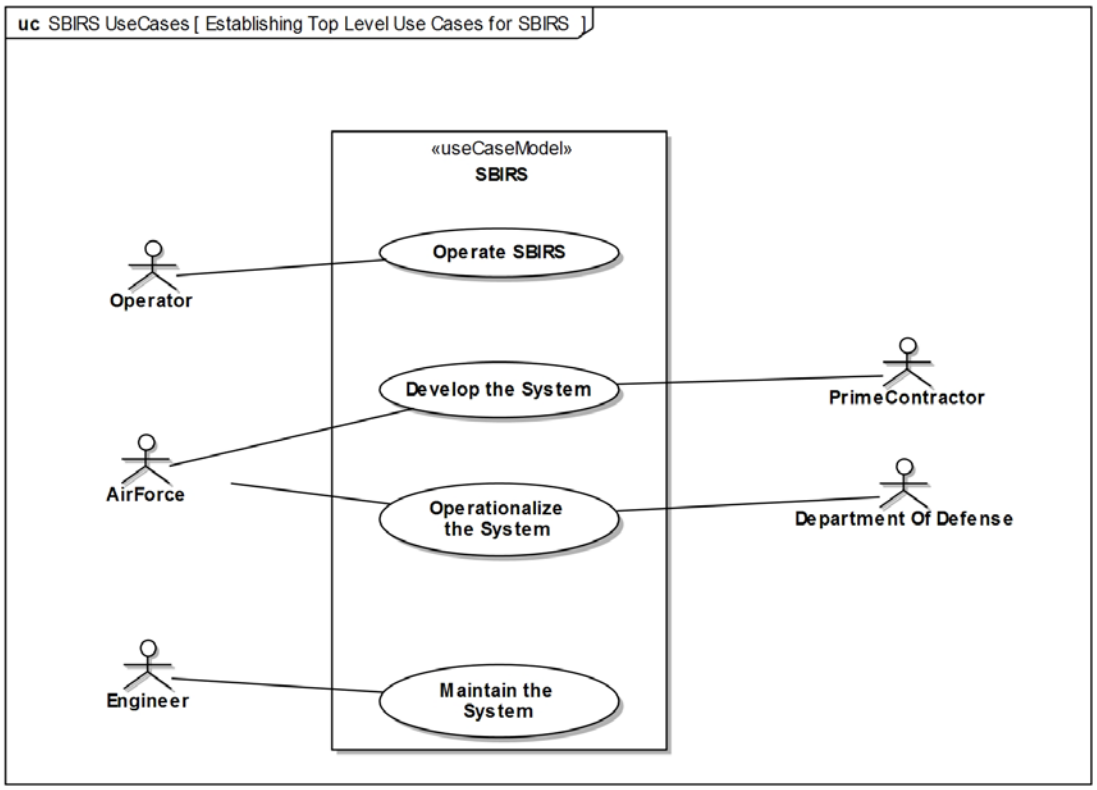

Figure 24. SysML Use Case Diagram Establishing the Top Level Use Cases for the SBIRS System Which Satisfies the OPIR Mission Area

3. Use Case Diagram—Operational Level

The hierarchical breakdown of the SysML use case diagrams introduced in Figure 24 is shown in Figure 25, specifically those associated with "Operate SBIRS." More precisely, this is a Goal-Level SysML use case diagram. The "goals" of the "fly the spacecraft" component of the "operate SBIRS" include initialize the spacecraft, maneuver, control payload, and collect and process data. The association of these goals to the use cases is shown as being "extended by" and "included within" the "fly the spacecraft" and "process and distribute data" use cases.

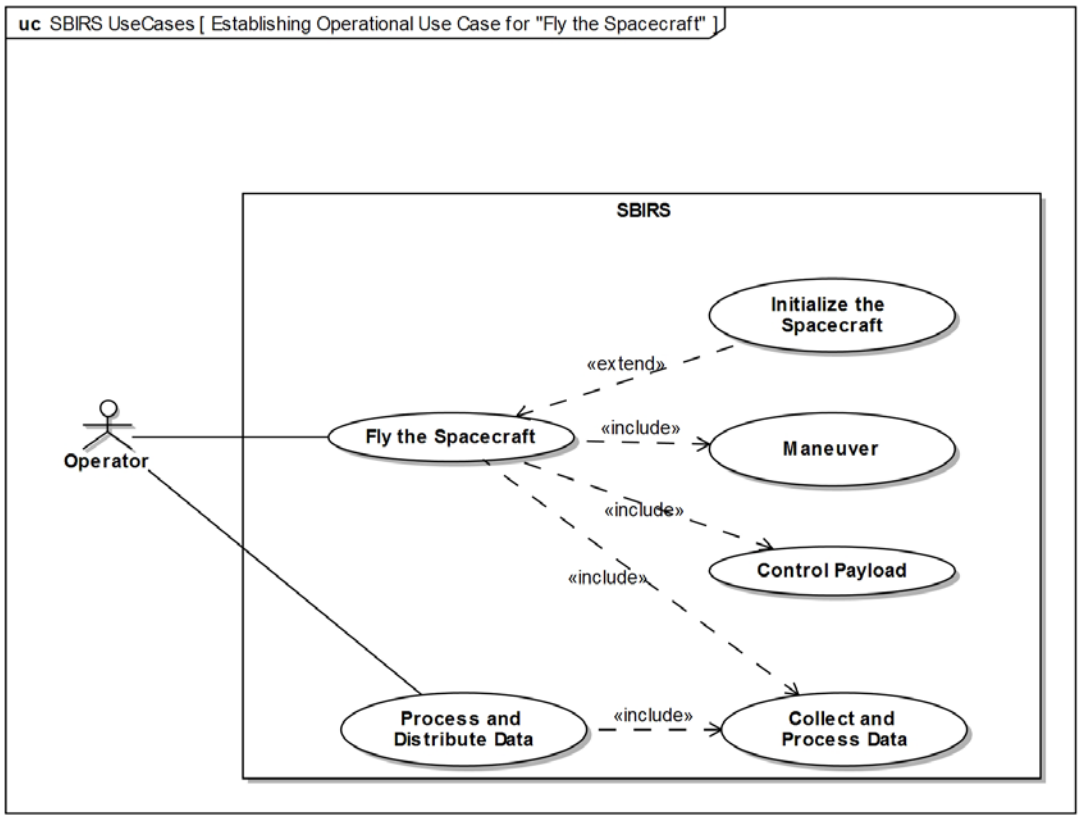

Figure 25. SysML Use Case Diagram Establishing the Operational Use Cases Which Further Refine the "Fly the Spacecraft" Use Case

4. Sequence Diagram—Initialize Black Box

The SysML sequence diagram in Figure 26 shows the interactions between the operator (actor) and the SBIRS (system) necessary for the "fly the spacecraft" use case introduced in Figure 25. At this level of abstraction, directionality of the interactions is not captured. Instead, this top-level sequence diagram provides insight into the sequencing and hierarchal interdependencies of lower level sequence diagrams. The term "black box" is used in this case to identify that the subject system, SBIRS, is interacting with outside external elements: the internal detail is not shown at this point in the model decomposition.

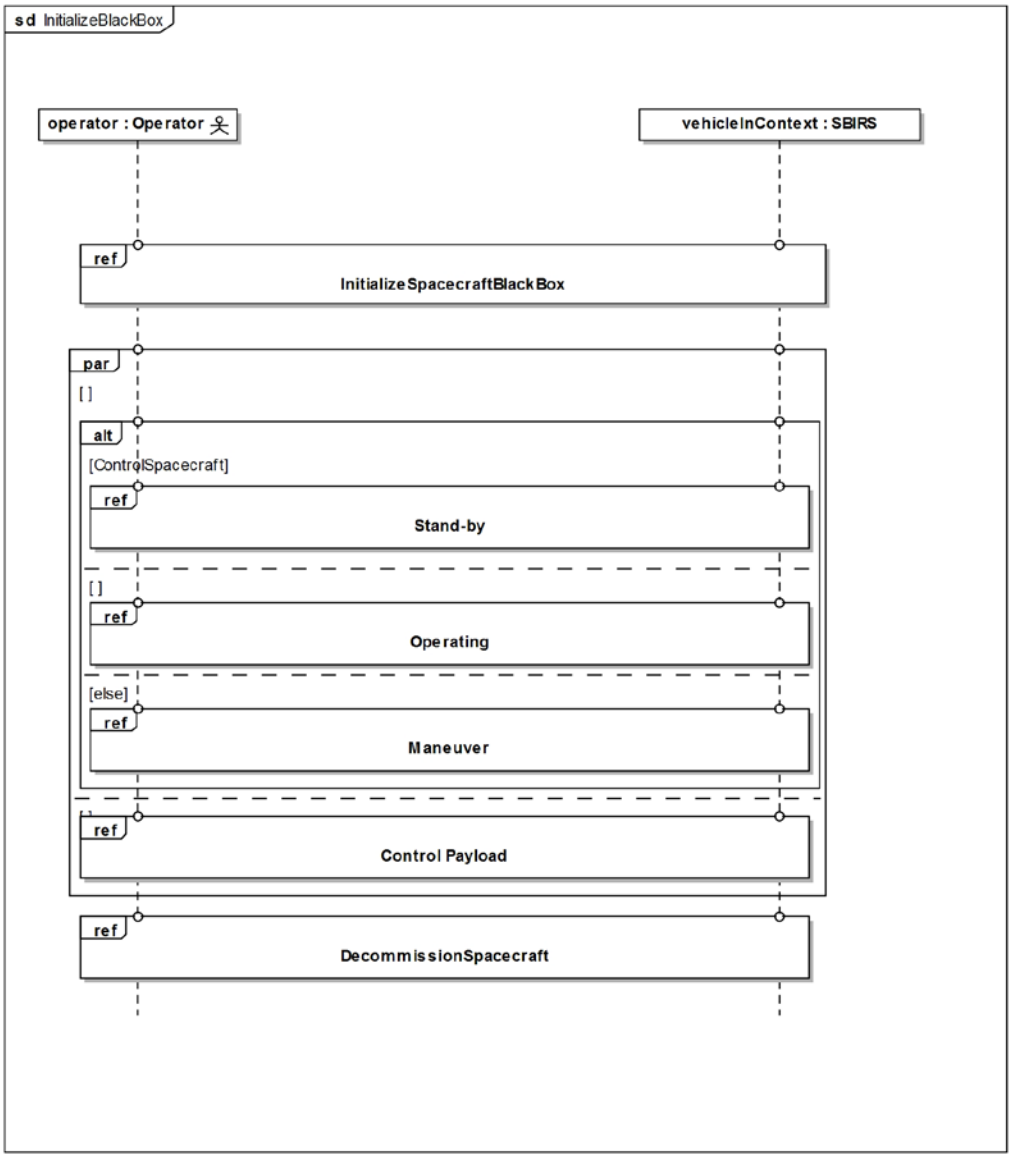

Figure 26. SysML Sequence Diagram Establishing the "Black Box" Top-Level Use Cases and Their Interdependencies

5. State Machine Diagram—Spacecraft Operational States

The SysML state machine diagram in Figure 27 identifies and describes the interaction between the operational states of the SBIRS system introduced as part of the black box sequence diagram shown in Figure 26. How these different operational states are triggered by the SysML model is shown later, as are the requirements that specify these operational states and the interactions and behaviors (e.g., drifting, activate,

acceleration) which trigger the system's transition from one state to another. The SysML diagram in Figure 27 introduces how requirements are allocated throughout the SysML model. As shown in Figure 27, the "Spacecraft Operational States" state machine diagram "refines" the "Power Source Management" requirement.

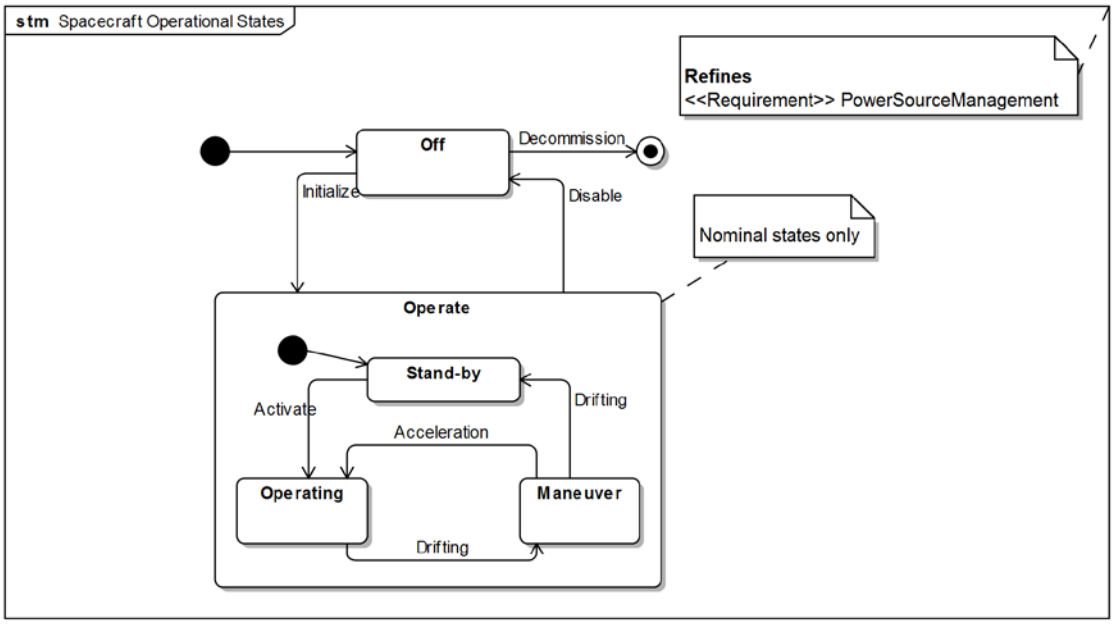

Figure 27. SysML State Machine Diagram Associated with the "Fly the Spacecraft" Use Case

6. Decomposed Sequence Diagrams

SysML sequence diagrams decomposing the top-level sequence diagram shown in Figure 26 are presented in Figures 28 and 29. The first of these decomposed sequence diagrams, shown in Figure 28, is the "initialize spacecraft black box," which shows the internal interaction, as defined by the "SendCommandToInitialize" and "InitializeSpacecraft" swimlines between the operator and the SBIRS system.

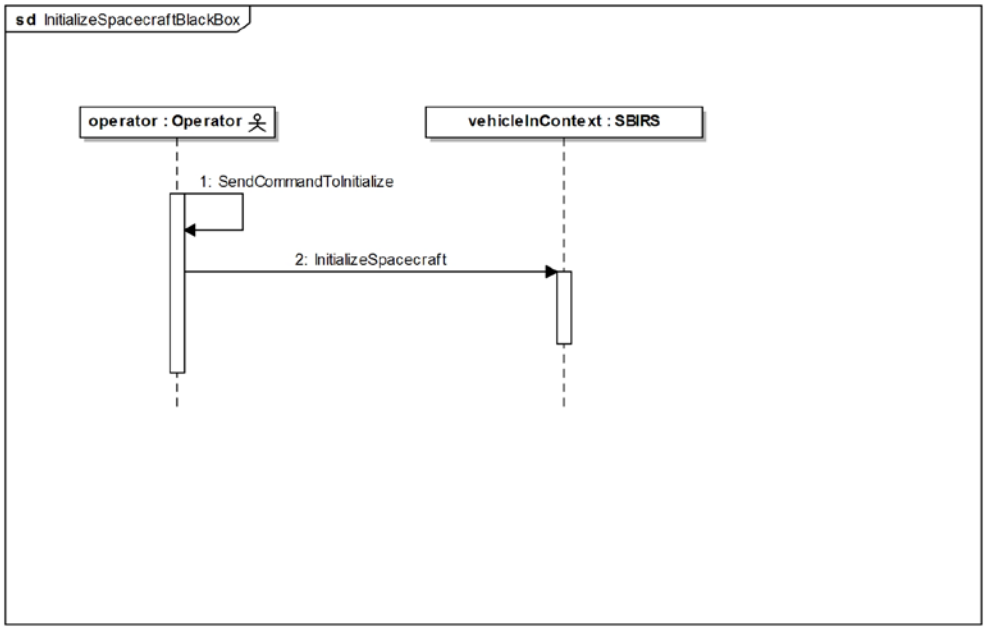

Figure 28. SysML Sequence Diagram Capturing the "Black Box" Interaction for the "Initialize Spacecraft" Use Case

The second decomposed sequence diagram, shown in Figure 29, presents an example of a "white box" sequence diagram detailing the interaction between the "Power Control Unit" and the "Electrical Power Controller," two subsystems of the "Power Subsystem," which is itself a sub-system to the SBIRS system of interest. This structural parts breakdown is shown later in Figure 37.

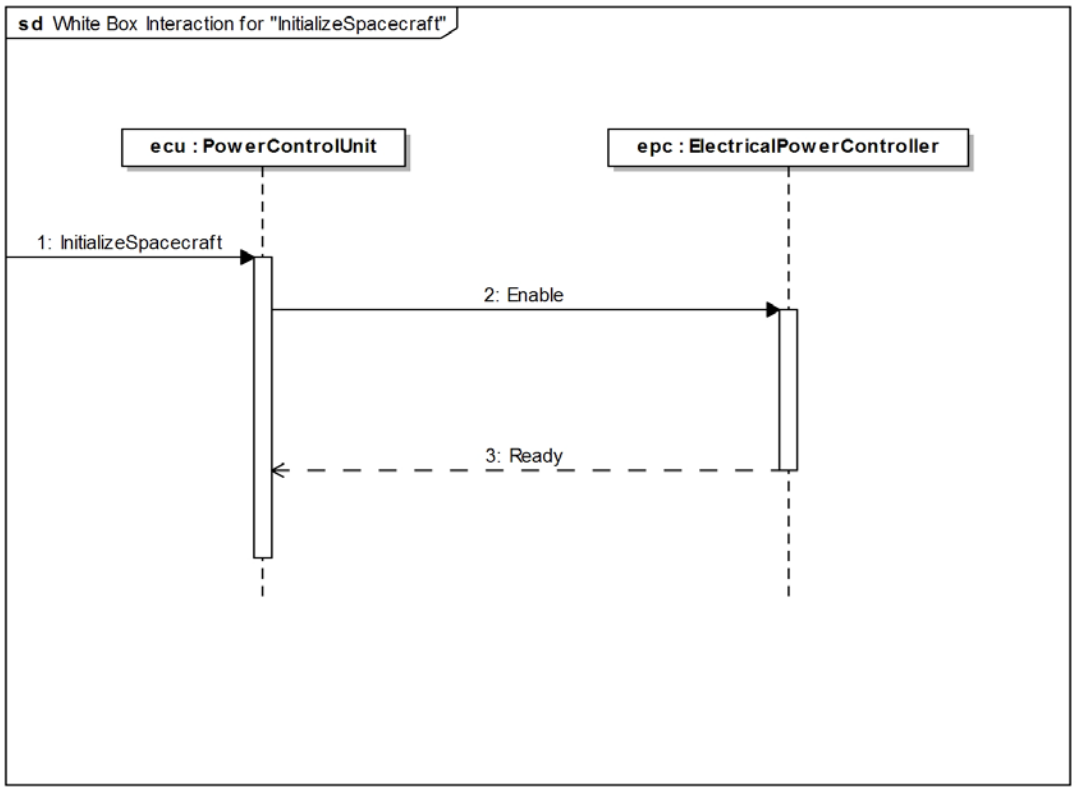

Figure 29. SysML Sequence Diagram Capturing the "White Box" Interaction for the "Initialize Spacecraft" Use Case

7. Requirements Diagrams

Three SysML requirements diagrams are shown in Figures 30, 31, and 32—capturing the SBIRS requirements hierarchy, derived requirements, and power system requirement relationships, respectively. Each of these requirements would be derived from the "OPIR Requirements Document" specification and input into the SysML model. The SBIRS requirements hierarchy shown in Figure 30 highlights some of these requirements. The "affordability" requirement is shown expanded to indicate the level of detail that can be captured on a SysML diagram. The SysML "containment" relationship (as shown by the cross hairs on one end of the relationship line) are used to show that a complex requirement is decomposed into simpler component requirements.

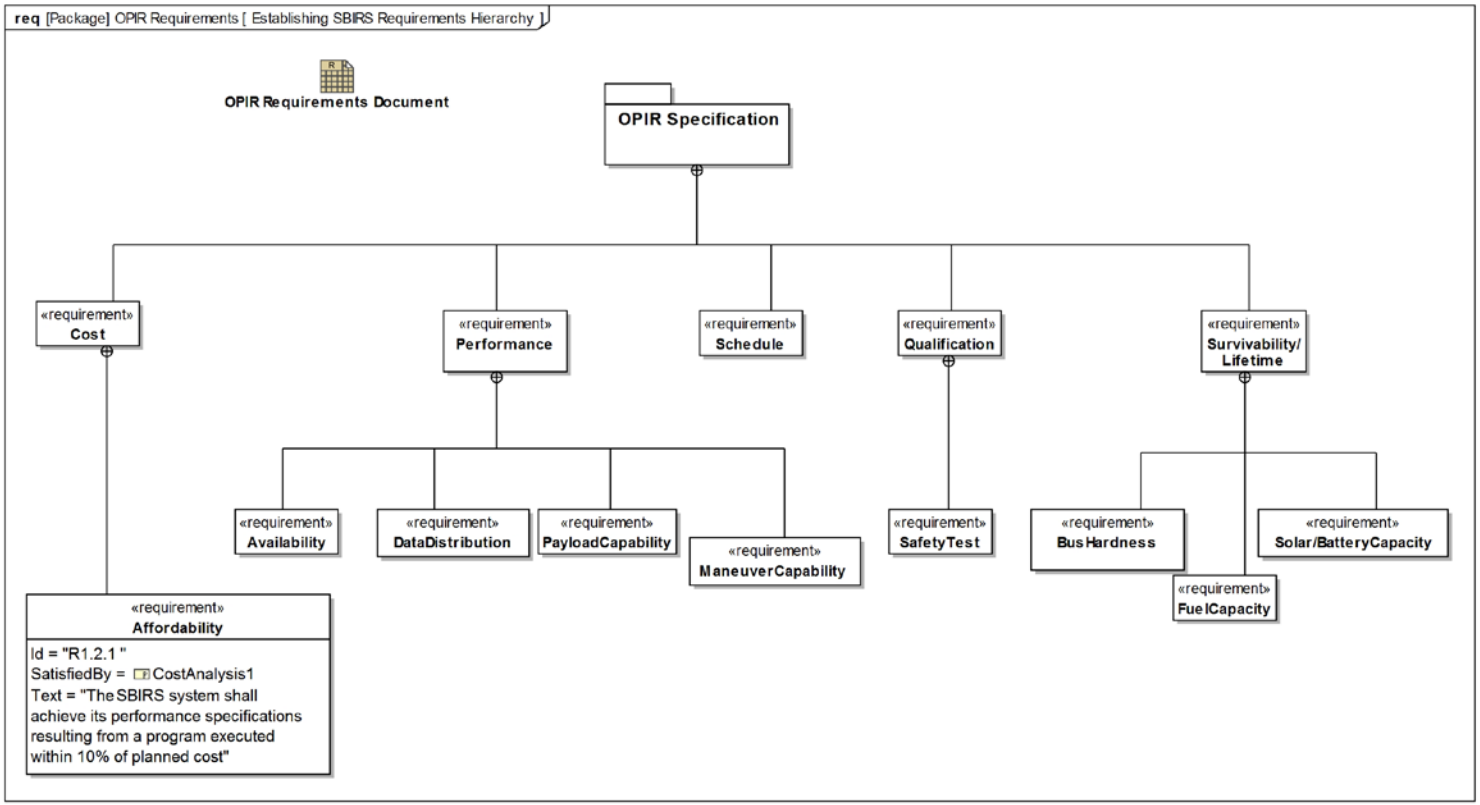

Figure 30. SysML Requirements Diagram Establishing the OPIR Requirements Hierarchy

The SysML requirements diagram shown in Figure 31 provides an example of what the requirements hierarchy might look like at the lowest level of the requirements derivation and decomposition within the model. The derived requirements, as indicated by the <<deriveReqt>> relationship, express the requirements in the OPIR Requirements Document specification in a manner that specifically relates them to the SIBRS system. This diagram shows the other end of the requirements allocation introduced by the state machine diagram in Figure 27 using the <<RefineBy>> relationship. Additionally, this SysML diagram introduces the use of the <<Rationale>> object that can be attached to any SysML relationship. In this case, the <<Rationale>> is attached to the relationship between the "Power System Loads" <<requirement>> and its <<derived requirement>> the "Power Source Management" <<requirement>>.

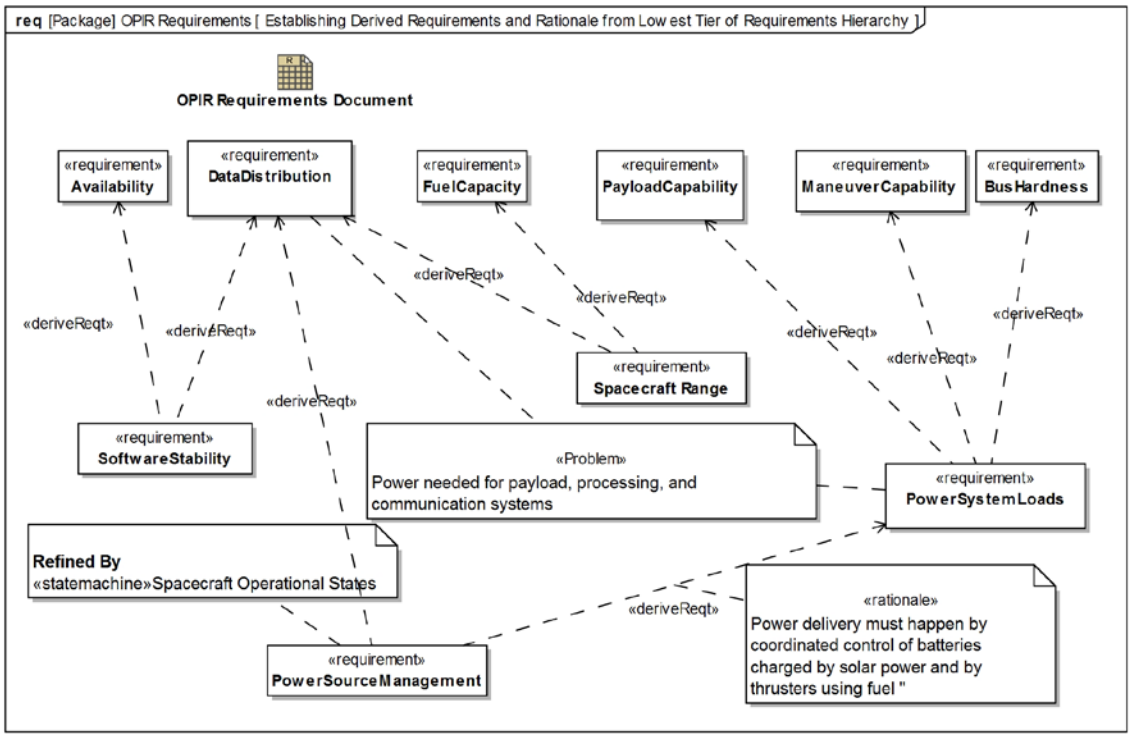

Figure 31. SysML Requirements Diagram Establishing the Derived Requirements and Rationale From the Lowest Tier of the Requirements Hierarchy

The third SysML requirements diagram shown in Figure 31 details the requirement relationships associated with the "maneuver capability" requirement. This

SysML diagram shows how the <<refine>> relationship can be used. In this case, it indicates that "maneuver capability" requirement is <<refined>> by the "maneuver" use case. This diagram also shows how a physical system component, such as the "power subsystem" can be shown to <<satisfy>> a requirement, in this case the "power system loads" requirement as shown in Figure 32 by the use of the <<satisfy>> relationship. Lastly, as will be elaborated later in this SysML case study, a <<test case>> class/block is identified and shown that it will be used to <<verify>> the "maneuver capability" requirement.

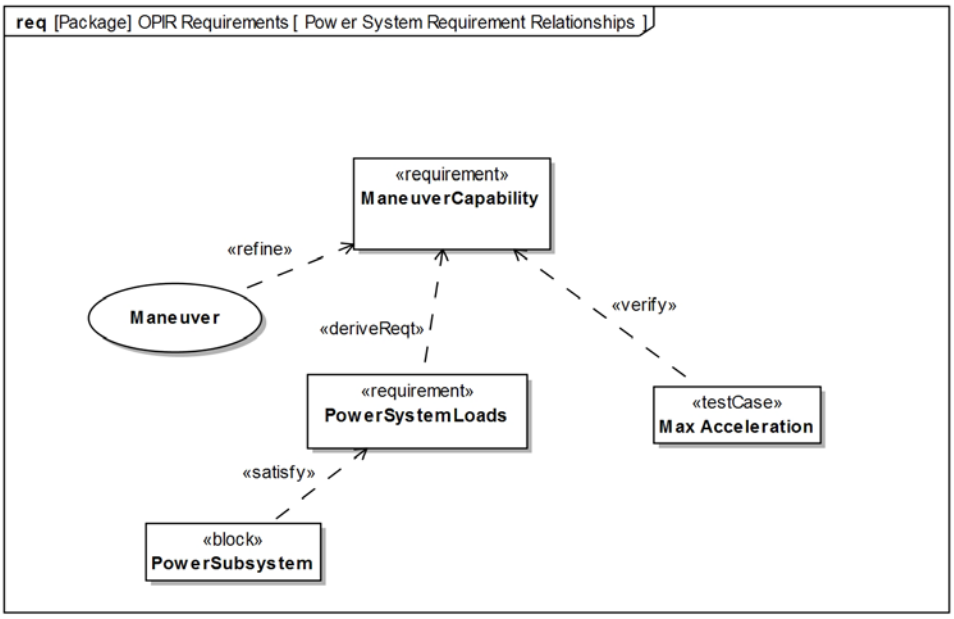

Figure 32. SysML Requirements Diagram Capturing the Relationships for the "Maneuver Capability" Requirement

8. Activity Diagrams

Figure 33 shows the top-level behavior of the "accelerate" function for the SBIRS system. Specifically, this diagram attempts to allocate the system-level behaviors to the "powersubsystem" which has the designed behavior of "providepower." However, as the comment in Figure 33 suggests, the systems engineer cannot achieve the appropriate

level of detail by modeling the behaviors at just this top-level and must further decompose the system behavior to fully identify those behaviors, which influence the "powersubsystem."

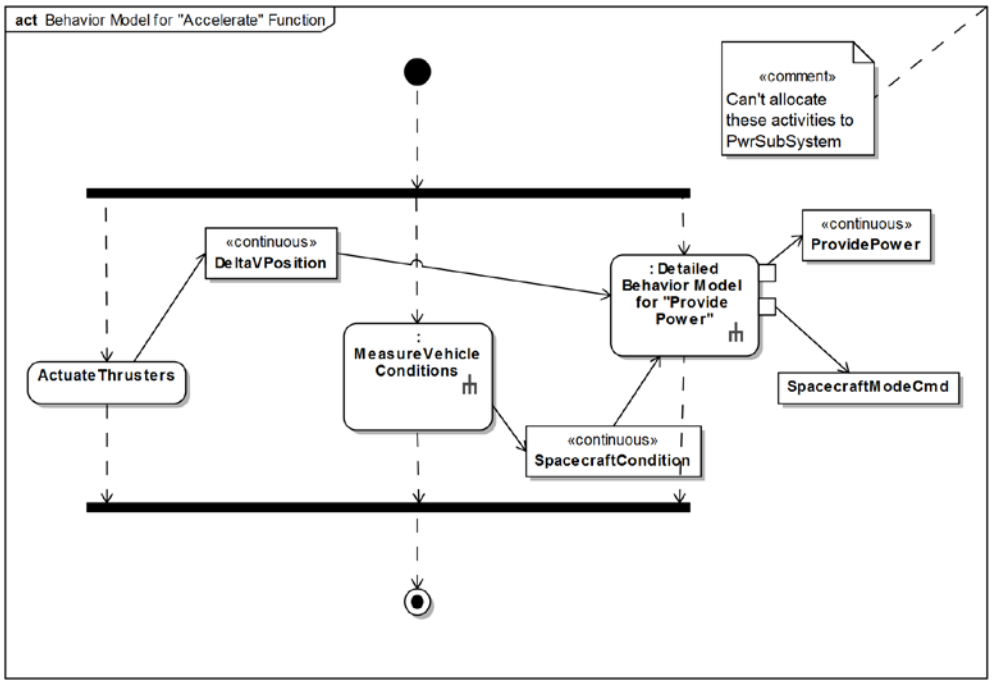

Figure 33. SysML Activity Diagram Highlighting the Behavior for the "Accelerate" Function

The top-level activities and object flows introduced in Figure 33 are further decomposed, as shown in the SysML block definition diagram in Figure 34.

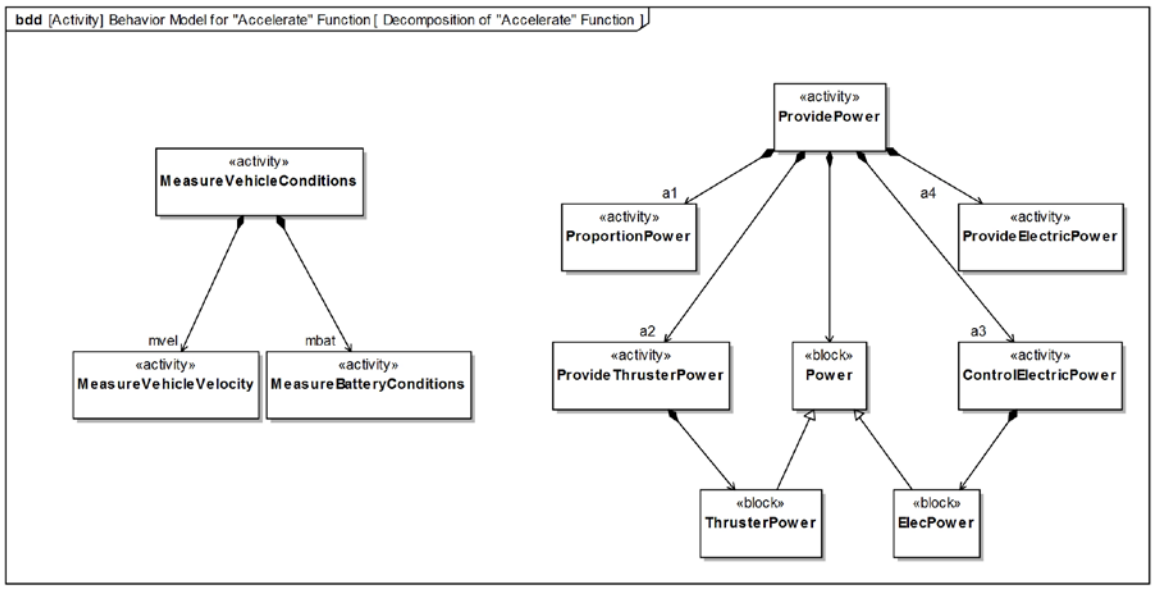

Figure 34. SysML Block Definition Diagram Decomposing the Activities Associated with the "Accelerate" Function

Given this de-composition of the top-level behavior associated with the "Accelerate" function of the SBIRS system, an activity diagram decomposing and detailing the "ProvidePower" activity is shown in Figure 35. This detailed SysML activity diagram includes the Actions which trigger the Activities and ObjectNodes introduced in Figure 34. The frame outlined by the vertical lines is an example of SysML AllocateActivityPartitions. These partitions provide insight into the allocation of the decomposed "ProvidePower" activities, including the "ProportionPower," "ProvideThrusterPower, "ControlElectricPower," and "ProvideElectircPower" activities to the physical system components/parts that must perform each of these activities, including the "PowerControlUnit," "PropulsionSystem," "ElectricalPowerController," and "SolarPannels," respectively. Similarly, one can see how the object flows "Throttle," "FilterPower," and "ElecCurrent" interact between physical system components and the activities performed by those components.

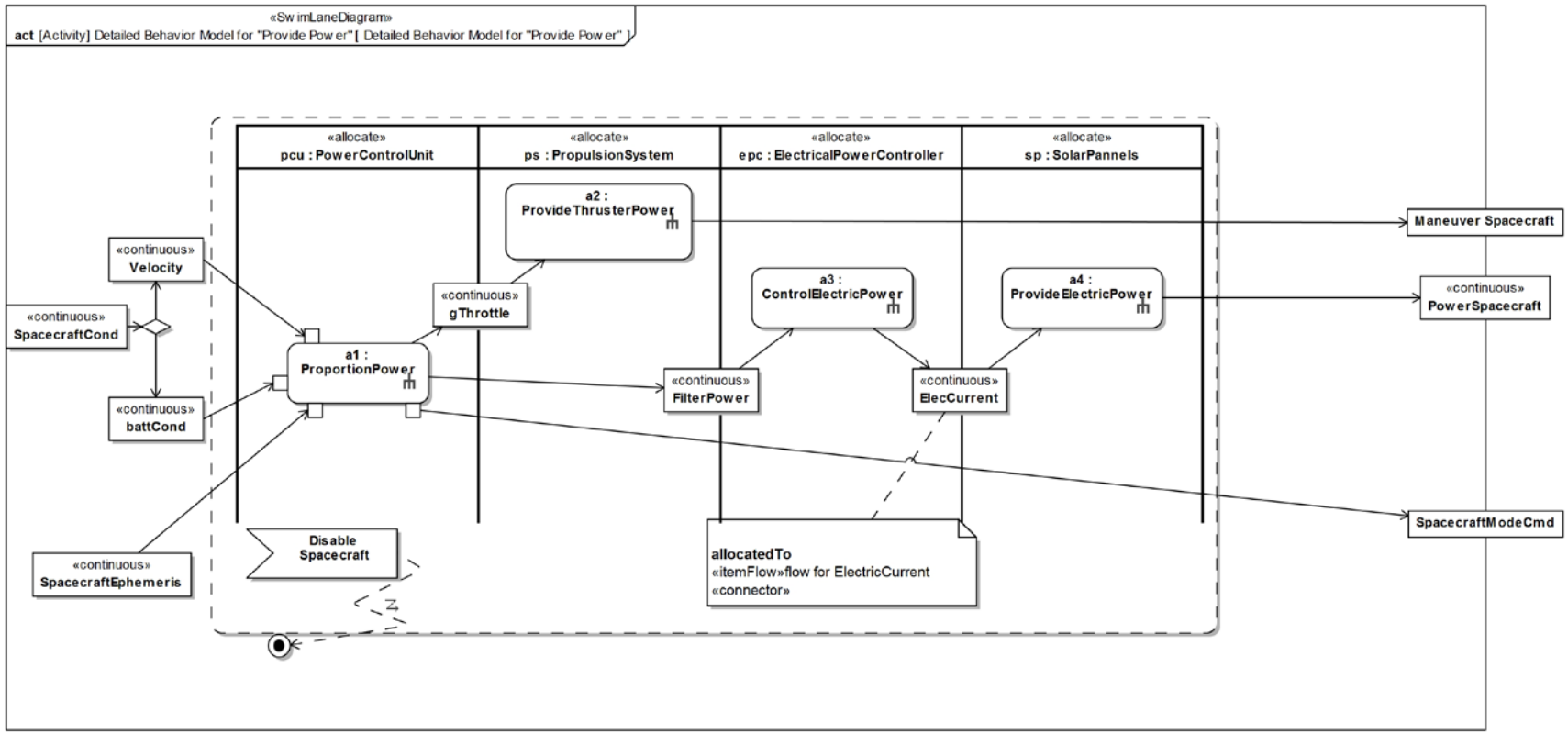

Figure 35. SysML Activity Diagram Providing a Detailed Behavior Model for the “Provide Power” Activity/Function

9. Block Definition Diagrams

The following diagrams provide examples of SysML block definition diagrams and internal block diagrams. The first of these block definition diagrams, shown in Figure 36, provides a refined decomposition and definition of the context diagram shown in Figure 23. This breakdown of the OPIR domain using the block definition diagram clearly specifies that the interactions "Initialize Black Box" and "Initialize Spacecraft Black Box," as shown in Figures 26 and 28, are owned by and therefore further refine the "OPIRDomain" block. The "1..*" identifier at the child end of the association between the "Environment" block and the ""Debris/OtherSystem" and "SystemOrbit" blocks indicates that there can be "one to many" instances of "Debris/OtherSystem" or "SystemOrbit" associated with the space "Environment."

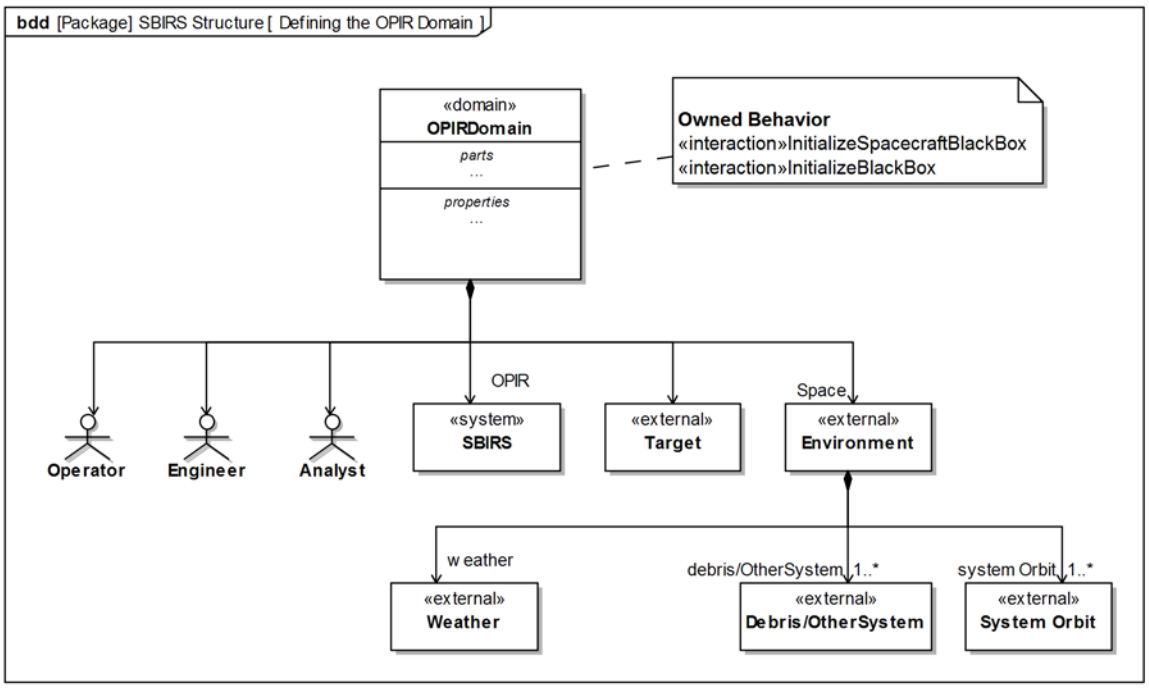

Figure 36. SysML Block Definition Diagram Defining the OPIR Domain

The block definition diagram shown in Figure 37 shows the top level decomposition of the SBIRS system into its physical subsystems. For illustration purposes, the "ThrusterSubsystem" and "StructureSubsystem" are further decomposed

(incompletely) into the “Thrusters” and “GyroAssembly” sub-systems, respectively. The solid-filled diamond shown at the parent end of the association between the “Thrusters” and the “ThrusterSubsystem” and the “GyroAssembly” and the “StructureSubsystem” in Figure 37 indicates a “composite aggregation,” or in other words, a “contained in” relationship, of the child, i.e., the “Thrusters” and “GyroAssembly,” to the parent, i.e., the “ThrusterSubsystem” and “StructureSubsystem.” The un-filled diamond indicates a “shared aggregation” between two parts/blocks. Therefore, it can be observed from Figure 37 that while the “Thrusters” and “GyroAssembly” parts are contained within the “ThrusterSubsystem” and “StructureSubsystem” blocks, respectively, both the “Thrusters” and the “GyroAssembly” are used by the “PowerSubsystem.” A <<rationale>> object can be added to further describe these types of dual-associations, as shown in Figure 37.

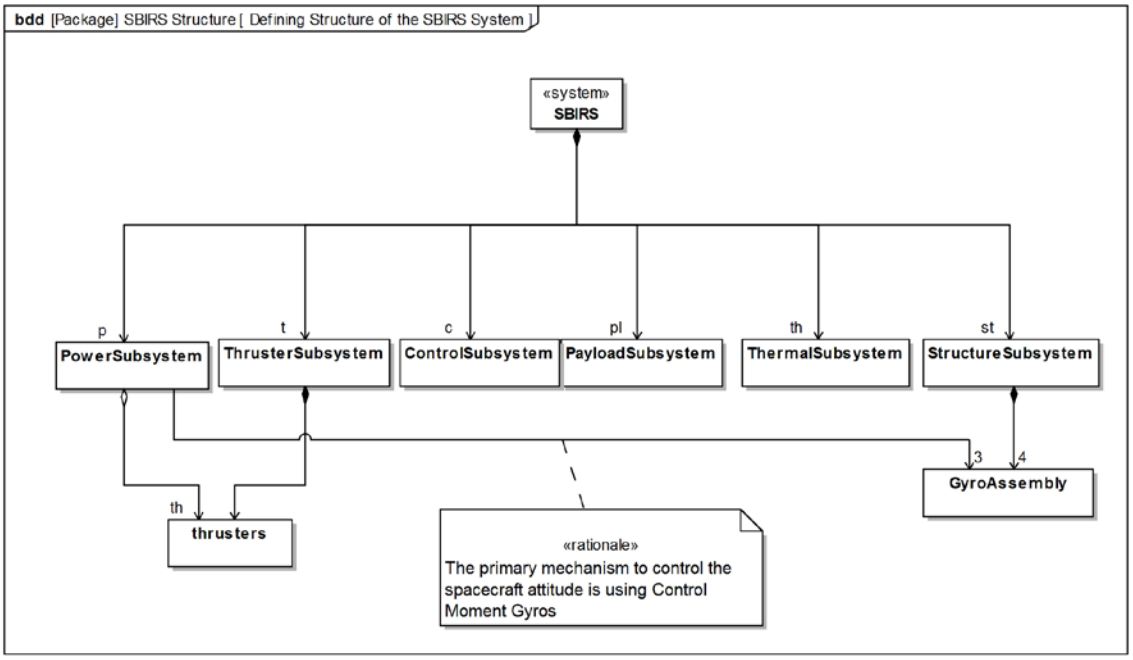

Figure 37. SysML Block Definition Diagram Defining the Structure of the SBIRS System

Figure 38 further defines the model elements introduced in Figure 37 by showing how they are connected together within the “SBIRS” block.

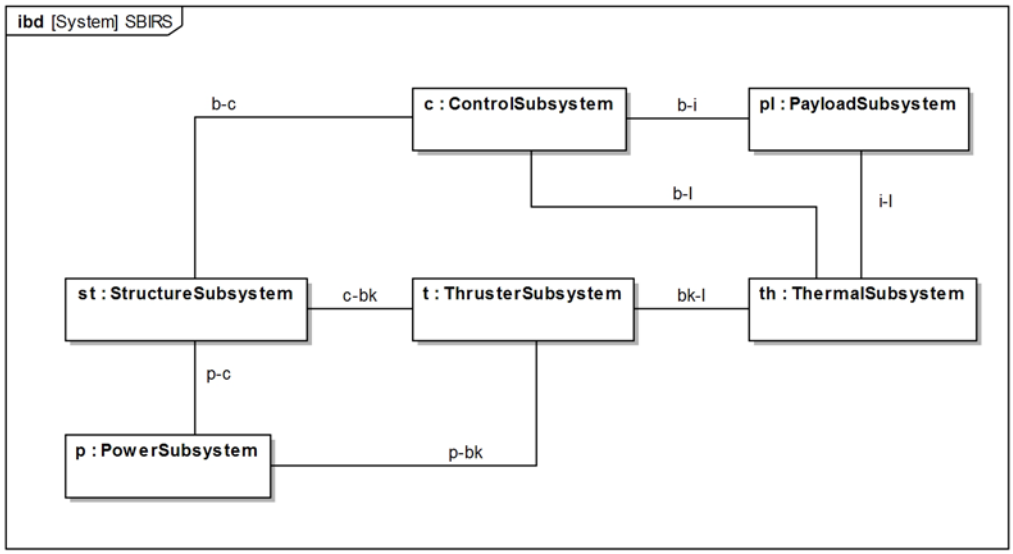

Figure 38. SysML Internal Block Diagram Capturing the Internal Structure of the SBIRS System

The "PowerSubsystem" introduced in Figure 37 is further decomposed into its sub-systems in Figure 39. This figure shows additional examples of the "use-not-composition" relationships between components, as indicated by the un-filled white diamond shape.

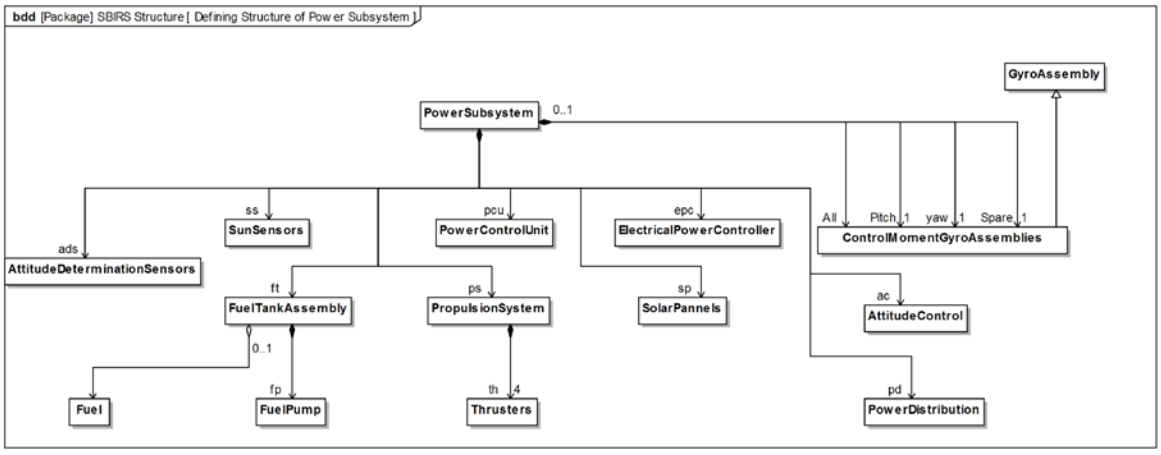

Figure 39. SysML Block Definition Diagram Defining the Structure of the Power Subsystem

While Figure 39 shows the defining structure of the “PowerSubsystem” component, it does not completely define how the components internal to the SBIRS power subsystem are used and how they interact and communicate with each other. The internal block diagram shown in Figure 40 does just this by defining the connectors between parts, ports, and connectors with item flows. The “use-not-composition” relationship of the “GyroAssembly” as defined in earlier diagrams is indicated in Figure 40 using a dashed-line border for the block. The ports are shown in Figure 40 as boxes with direction arrows placed either on a block or on the border of the diagram, indicating flows external to the diagram. An example of an item flow between ports is shown in Figure 40 as the fuel flow between the “FuelPump” and the “Thrusters.” “Required and provided” interfaces specifying each block are indicated by the ball and socket icons extended from some of the ports shown in Figure 40.

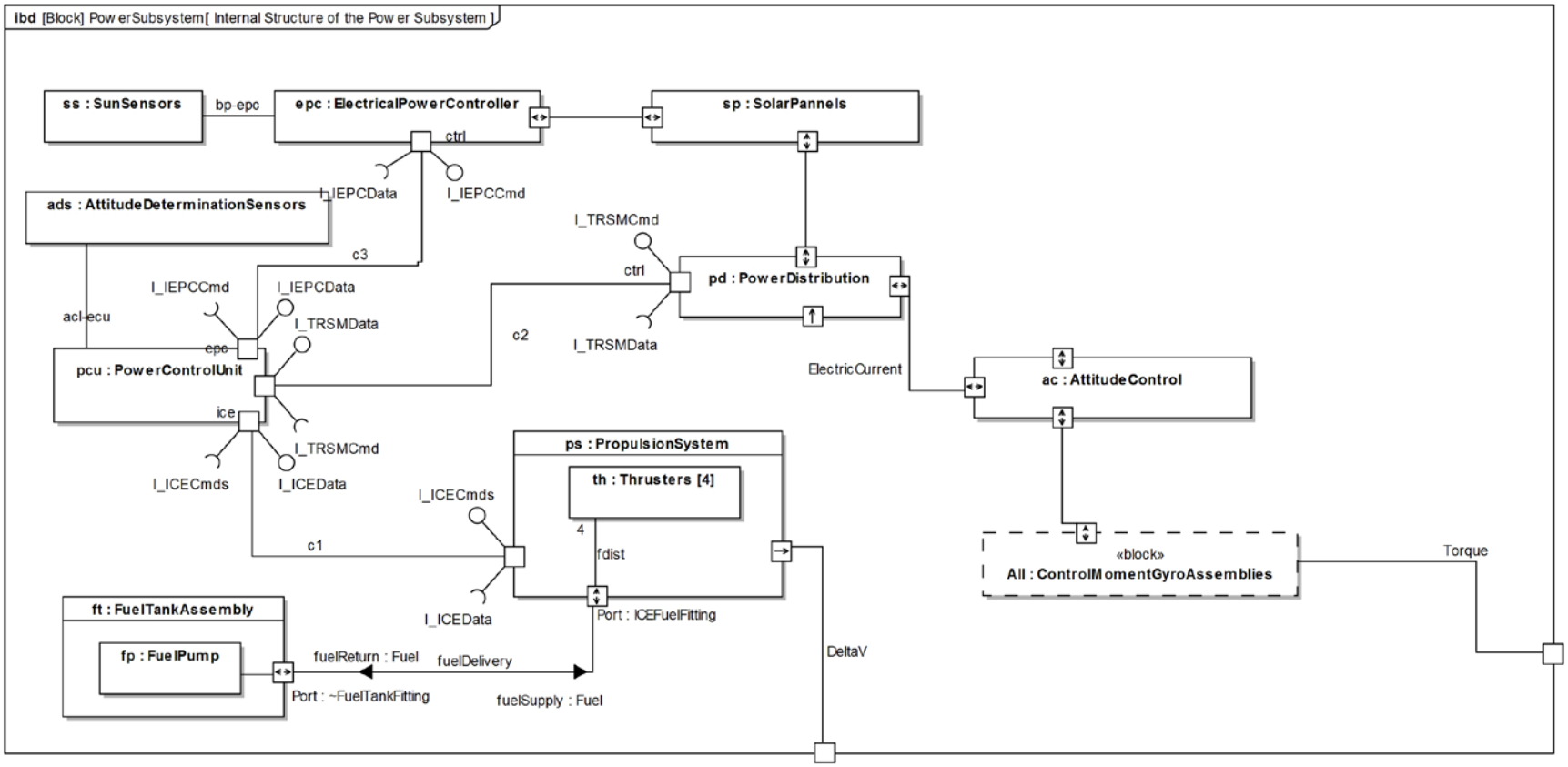

Figure 40. SysML Internal Block Diagram Defining the Internal Structure of the Power Subsystem

Another example of an internal block diagram is shown in Figure 41. This internal block diagram within the "PowerSubsystem" further refines the Controller Area Network (CAN) bus architecture using the ports defined earlier.

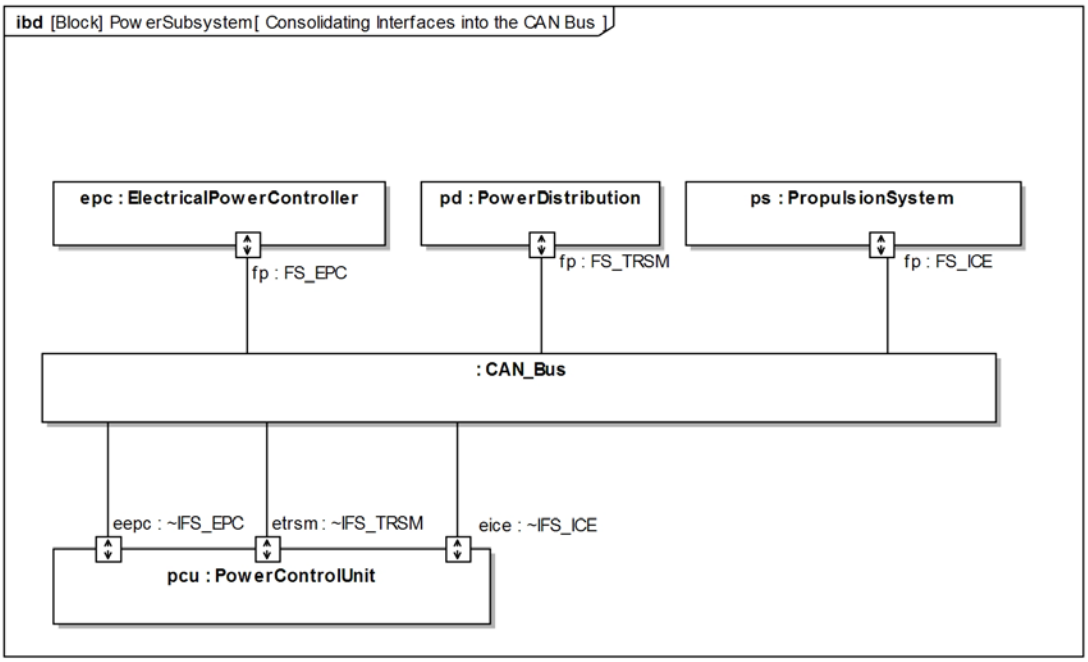

Figure 41. SysML Internal Block Diagram Identifying the Connectors into the CAN Bus

The explicit structural allocation between the connectors introduced in Figure 41 is further defined in Figure 42.

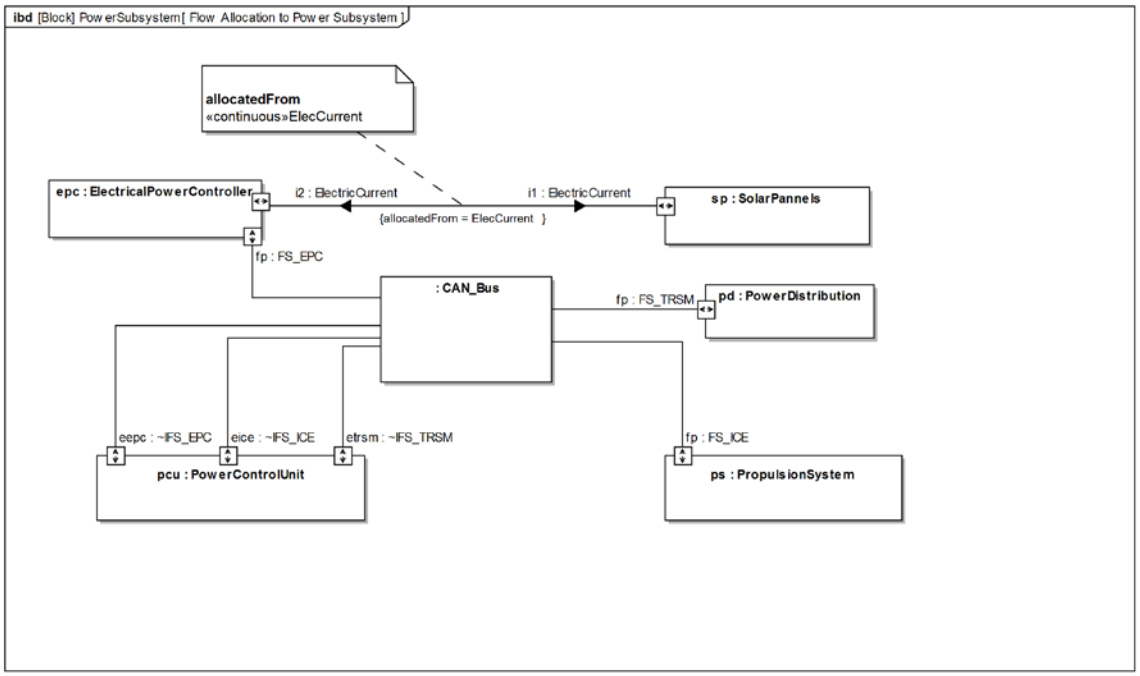

Figure 42. SysML Internal Block Diagram Detailing the Flow Allocation to the Power Subsystem

The port identified in Figure 40 for the "FuelTankAssembly" and "PropulsionSubsystem" along with the fuel flow between the two ports is further defined in Figure 43. This diagram defines the specific properties of the "FuelFlow," including the "fuelReturn" and "fuelSupply" item flows first introduced in Figure 40. Figure 43 also identifies the measured and observed properties, pressure and temperature, of the fuel itself within the "Fuel" block.

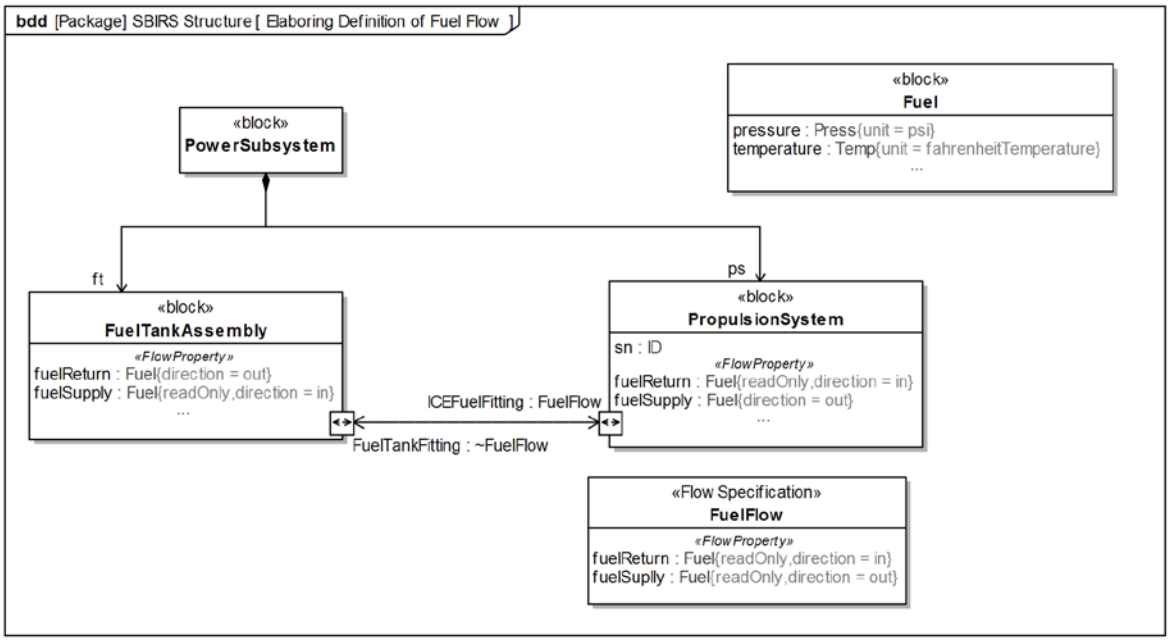

Figure 43. SysML Block Definition Diagram Detailing the Definition of "Fuel Flow"

Figure 44 expands the "FuelTankAssembly" and "PropulsionSubsystem" blocks and further defines the "fuelDelivery" and "fdist" connectors introduced in Figure 40. The expansion of the "PropulsionSubsystem" reveals the "FuelRegulator" and "FuelRail" parts. These sub-system component parts are related to the original components through an allocation relationship, as indicated by the "allocatedFrom" box. Furthermore, it can be observed in this internal block diagram that the "fuelDelivery" connector is really two distinct connectors, "fuelSuppy" and "fuelReturn." The "Fuel" block within the "FuelTankAssembly" represents a quantity of fuel remaining. This fuel is drawn from the "Fuel" block to the "FuelPump," from which it is provided to the "PropulsionSystem" via the "fuelSupplyLine," as shown in Figure 44. The "fuelReturnLine" flow indicates that an un-used fuel can be returned to the "Fuel" block for later use.

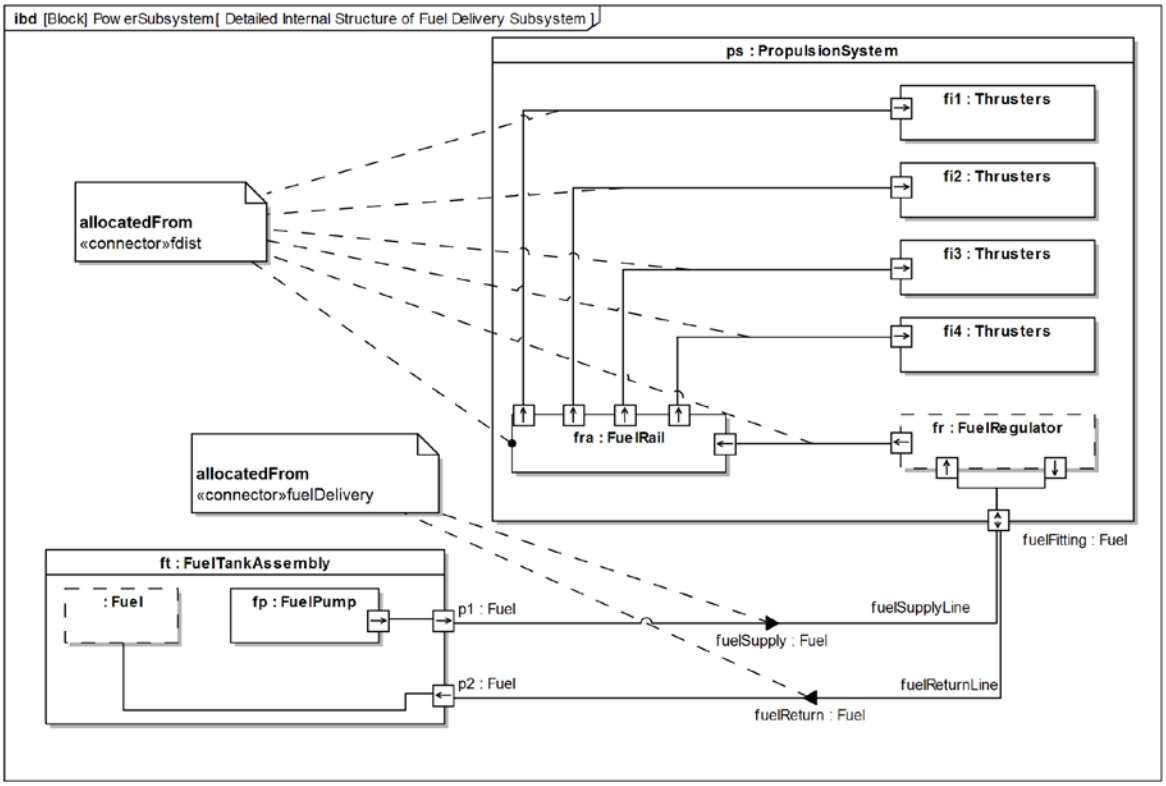

Figure 44. SysML Internal Block Diagram Detailing the Internal Structure of the Fuel Delivery Subsystem

10. Parametric Diagrams and Performance Analysis

Figure 45 introduces the SysML parametric diagram. Specifically, Figure 45 defines precisely how the fuel flowrate is related to "fuelDemand" and "fuelPressure." These relationships set the stage for modeling the precise engineering physics and behaviors of the item flows between the ports, connecting components within the power subsystem, a component with the SBIRS system. Such detailed and comprehensive definition of the system architecture, with equations precisely defining the relationships, can later be used to perform modeling and simulation in support of trade studies between multiple concept system architectures.

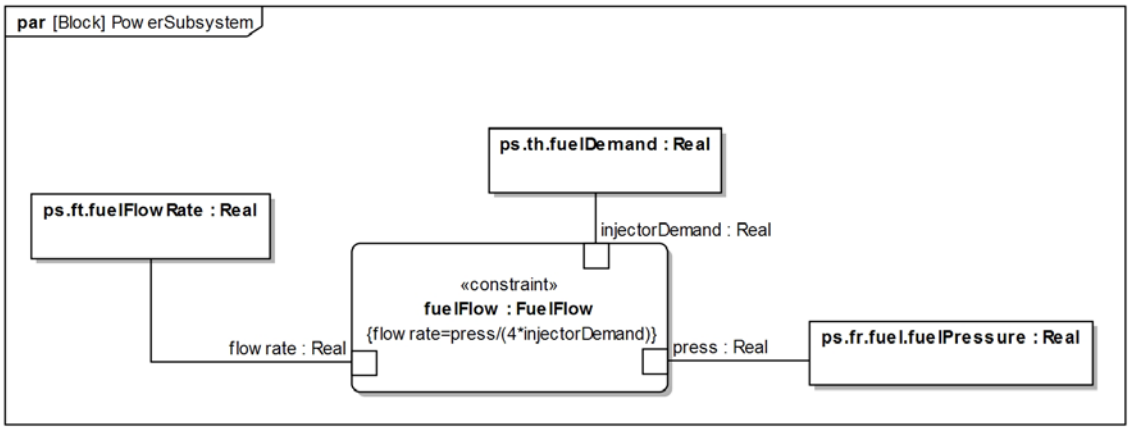

Figure 45. SysML Parametric Diagram Defining the Fuel Flow Constraints

The following diagrams further illustrate how SysML can be used to perform such an engineering analysis. The various engineering equations (along with their relationships to the domains and contexts previously introduced) that will be used to conduct an analysis of the propulsion sub-system are shown in Figure 46. These equations are modeled as <<constraint>> blocks within SysML, as shown.

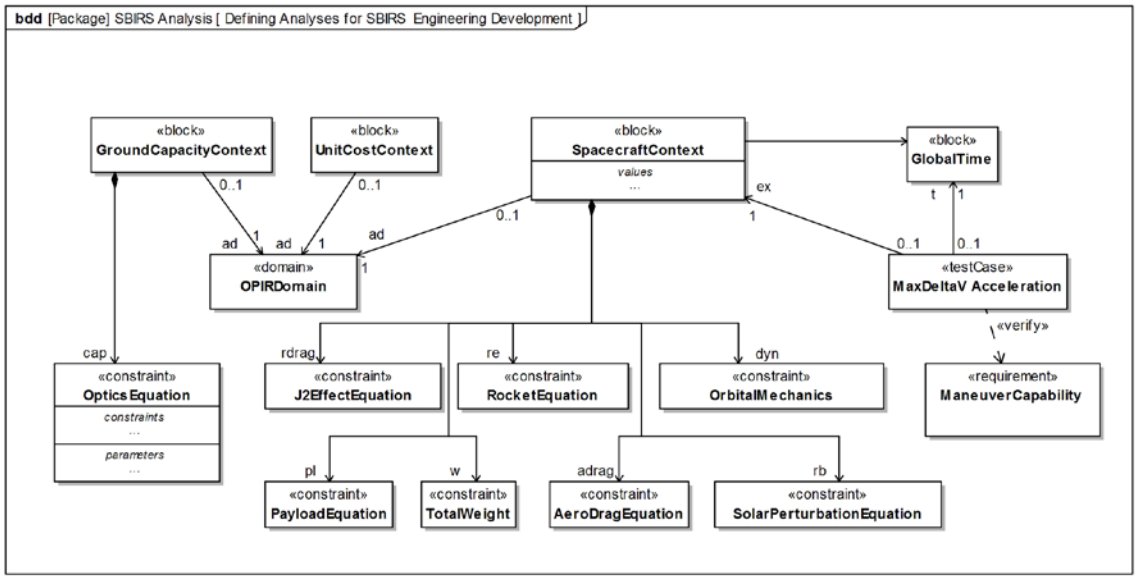

Figure 46. SysML Block Definition Diagram Defining the Analysis for the SBIRS Engineering Development

Figure 47 depicts a SysML package diagram, which provides additional detail of the user-defined Performance Viewpoint and the elements that populate the SBIRS specific "PerformanceView." The specifications of the Performance Viewpoint itself are identified using the SysML <<view point>> block. Many SysML diagrams support the "PerformanceView" representation of the Performance Viewpoint, including the "Operator" actor, "Fly the Spacecraft" use case, and "Performance" requirement, all of which were introduced and defined in greater detail earlier in this chapter. Figure 47 also introduces the measures of effectiveness (MOEs) that will be used as part of the engineering analysis of this particular propulsion sub-system architecture as part of the SBIRS system.

The performance viewpoint shown in Figure 47 is analogous to the "operational viewpoint" defined by the DoDAF, as previously discussed. This is one example of how SysML is something that can be related to and is compliant with DoDAF 2.0.

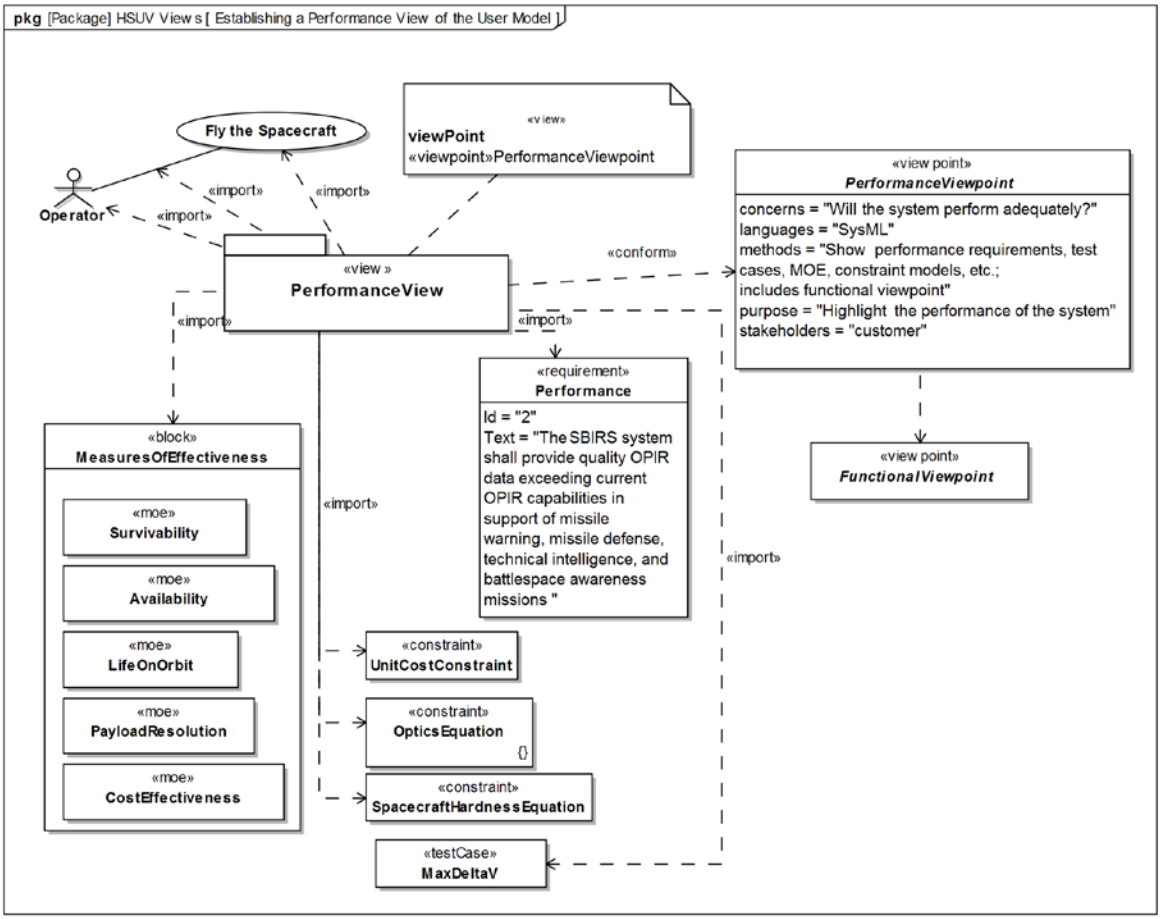

Figure 47. SysML Package Diagram Establishing the Performance View and Viewpoint of the OPIR Model

The MOEs introduced in Figure 47 are further defined in the SysML parametic diagram shown in Figure 48. This diagram describes how the overall cost effectiveness of the particular propulsion sub-system design alternative is evaluated against each of the defined MOEs. Figure 48 also introduces the <<objectiveFunction>> that is used to measure and compare each design alternative as part of an analysis of alternatives or trade study. In order to provide consistency between the analysis of various design alternatives, the same equations, objective function, and MOEs along with the relationships between each as shown in Figure 48 must be used for the analysis of each alternative.

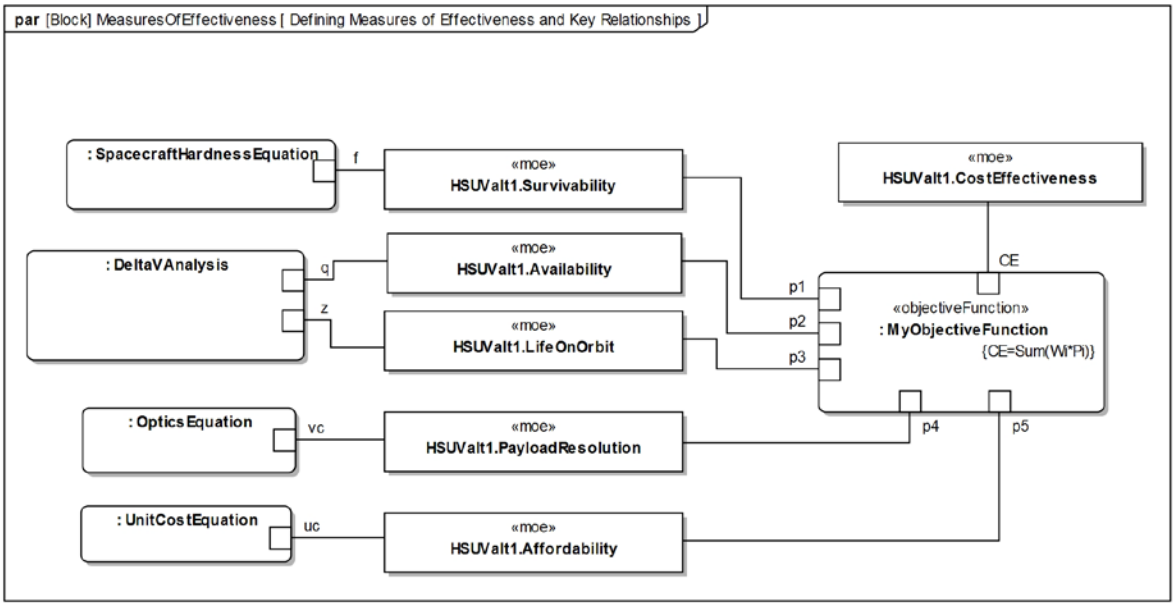

Figure 48. SysML Parametric Diagram Defining the Measures of Effectiveness and Objective Function for Engineering Analysis

One of the most significant life-limiting factors of any spacecraft is the on-board fuel. As such, the efficiency of a spacecraft's use of its fuel to perform activities such as station-keeping maneuvers is a key requirement to the design of the propulsion sub-system. In order to assess the efficient use of fuel by the propulsion sub-system specified for this example, we can examine the constraint blocks and properties necessary to evaluate the fuel efficiency presented in Figure 49. The equations and interactions shown in Figure 49 establish the mathematical relationships for the fuel efficiency calculations.

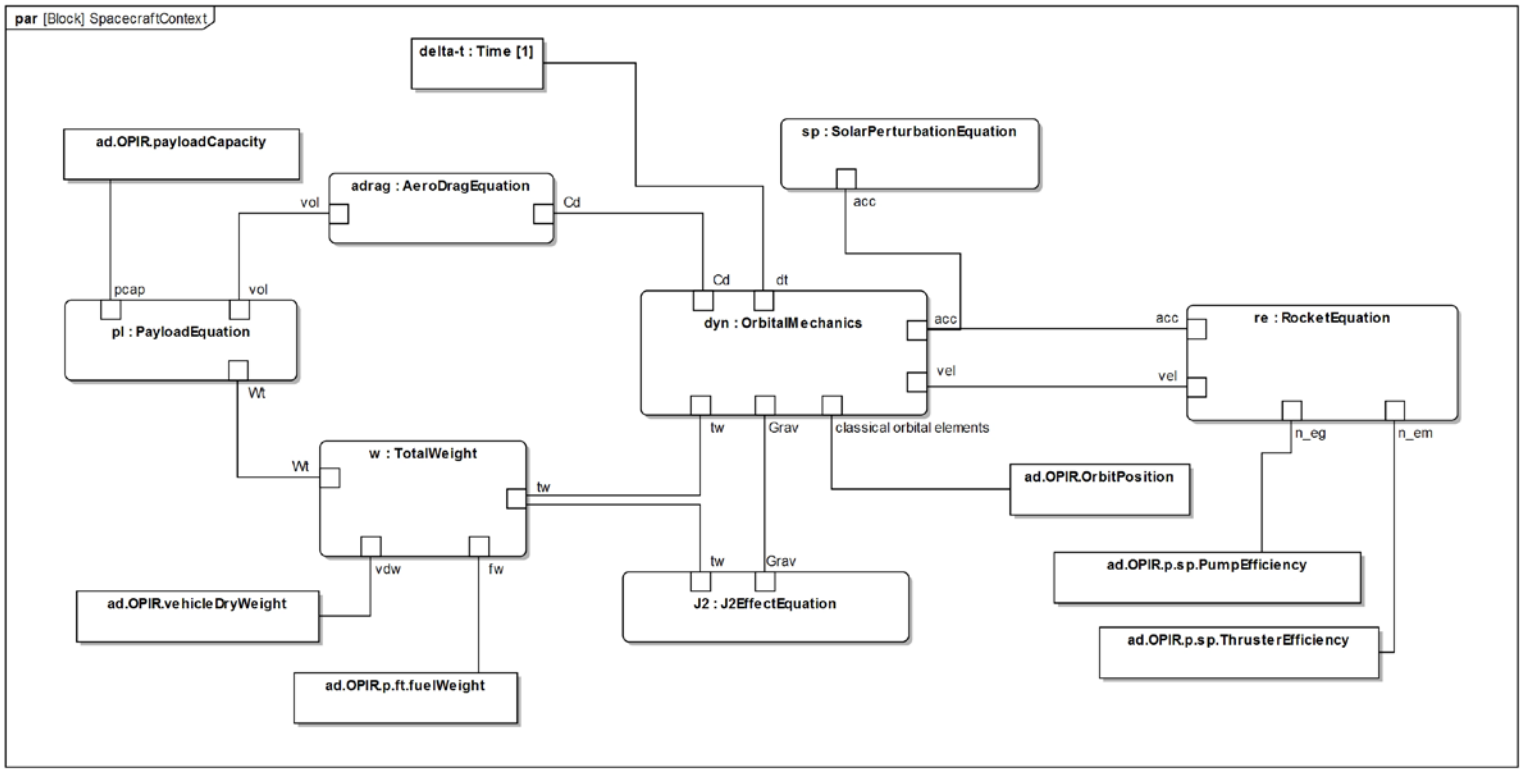

Figure 49. SysML Parametric Diagram Establishing the Mathematical Relationships for Analysis

The "OrbitalMechanics" constraint block introduced in Figure 49 is further decomposed in Figure 50. This diagrams shows the use of SysML Constraint Nodes, identifying the equation associated with each constraint within the {} brackets. <<rationale>> blocks can also be used to provide a visual of the equations used for each constraint, as shown. As with other SysML diagrams, the internal parametric diagram shown in Figure 50 shows the inputs and outputs of the parent "OrbitalMechanics" block as entering from the left side of the diagram and exiting the right side of the diagram.

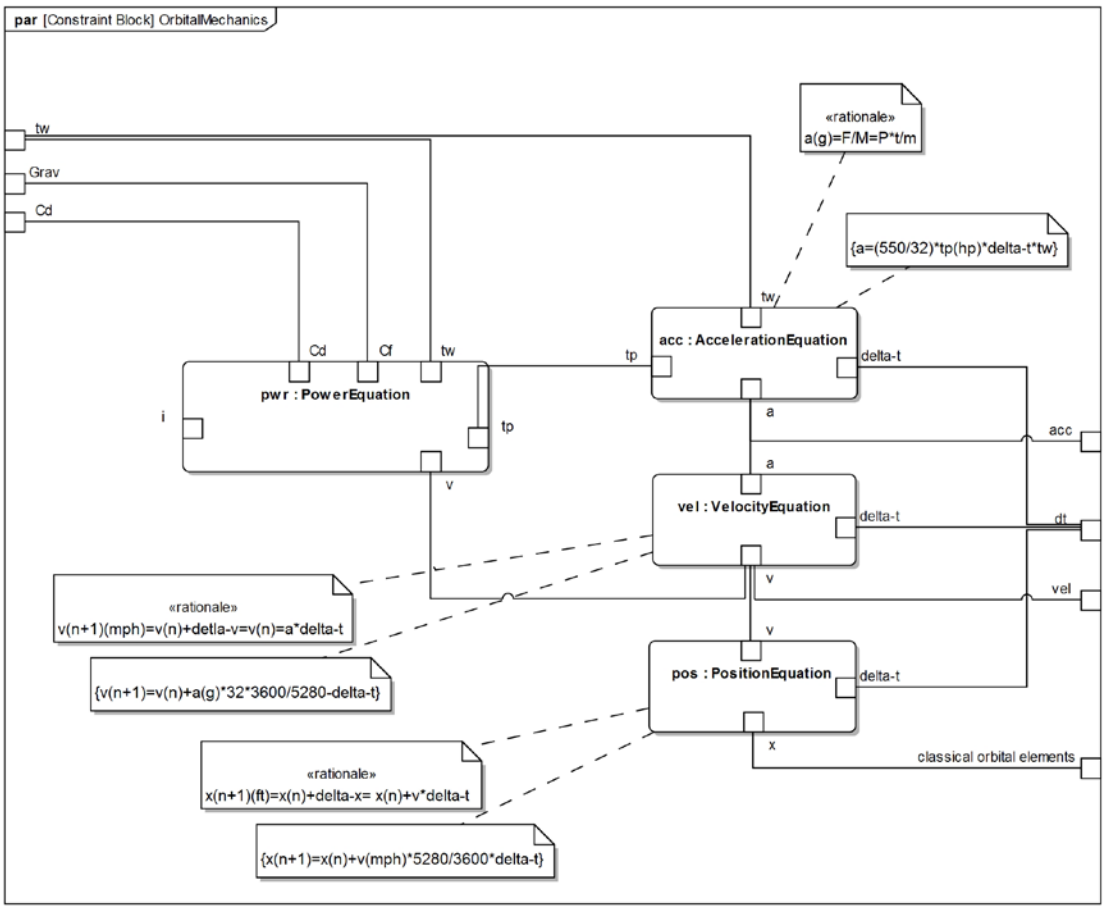

Figure 50. SysML Parametric Diagram Detailing the "Orbital Mechanics" Mathematical Model

Utilizing the engineering constraints applied to components throughout the SBIRS architecture decomposition, the SysML architecture becomes a powerful

executable model that can be used by designers, systems engineers, and ultimately decision makers to perform Modeling and Simulation and Analysis of Alternatives or trade studies comparing the various system designs and implementations that satisfy the architecture. An example of a report that can be generated from this executable model is shown in Figure 51 for the "Maximum Delta-V Acceleration Analysis" conducted against the propulsion sub-system of the SBIRS system using the architecture and associated engineering equations/constraints outlined in this case study. As indicated by the SysML timing diagram in Figure 51, this particular analysis satisfies the "ManueverCapability" requirement as shown in Figure 32.

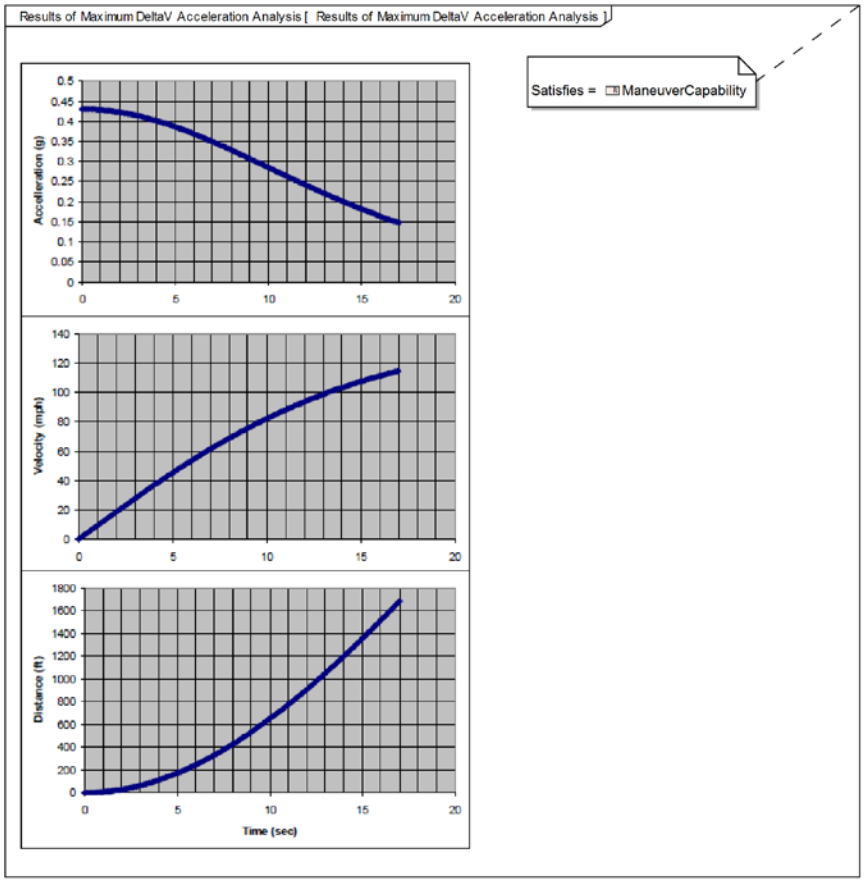

Figure 51. SysML Timing Diagram Showing Sample Results from a SysML Parametric Analysis. This Example Summarizes Results from the Maximum Acceleration Analysis

Additional parametric models and engineering analysis can be generated based on the unique requirements of the system being modeled. All results from lower-level engineering analysis, such as the acceleration analysis summarized in Figure 51, can then be rolled up through each level of the system architecture de-composition and ultimately result in top level performance metrics for a particular system of interest. By analyzing many different system concepts this way, measured against the same architecture de-composition and SysML model, the assessed performance of each system concept can be compared against the cost and other programmatic factors of interest. Then, using analysis tools such as cost-effectiveness comparisons, as described in Chapter III: Section D of this paper, decision makers can make truly educated decisions based on a rigorous engineering analysis.

The SysML diagrams developed for the SBIRS system implementation to the OPIR Mission Area Architecture illustrates how SysML, combined with the concepts of model based systems engineering, can provide extremely powerful engineering tools in support of systems engineering. Examples of some of the commonly asked questions that could be answered with relative ease and great fidelity once a structured SysML model is complete are provided below:

- What orbital regimes (inclination, altitude, etc.) and at what note (geographical longitude of the ascending note) should the SBIRS spacecraft be placed in to optimize coverage area? Revisit time? Dwell time? Sensor resolution?
- How many spacecraft are required to achieve mission requirements?
- What Infrared payload detector material should be used to optimize payload performance?
- What size payload telescope should be used to optimize payload performance?
- How much fuel is required to sustain the spacecraft for a required duration?
- What is the overall reliability of the SBIRS spacecraft?
- What structural, thermal, and other impacts does the power subsystem inflict on the thermal control subsystem? The structure/spacecraft bus subsystem?

Ultimately, defining the system architecture by answering all of the questions above, an executable SysML architecture could be used to answer the question: How well does any particular architecture, defined by decisions made to the questions above, satisfy the mission requirements (missile warning, missile defense, technical intelligence, and battlespace awareness)? MBSE is a powerful part of helping to answer this question by providing all stakeholders, from the design engineers and architects all the way up to the highest level decision maker, with detailed products (models and views). These MBSE products provide stakeholders with easily digestible insight into the robust engineering analysis conducted by the SysML architecture framework and can be used to enable highly informed and effective decision making based on detailed analysis.

E. ARCHITECTURE DEVELOPMENT—HEURISTICS

In effort to outline basic guidelines for developing a system architecture, such as the OPIR MASA and the SBIRS system implementation of this architecture shown in this SysML case study, the author of this report has developed ten heuristics; these heuristics have been developed through the writing of this report as well as from interactions with other systems, and they are provided in Appendix A.

The Merriam-Webster dictionary defines heuristics as "aids to learning, discover[ing], or problem-solving by experimental and especially trial-and-error methods" (Merriam-Webster Dictionary, under "heuristics" 2013). Another term for heuristics, used often in the Department of Defense acquisition community, is to say "lessons learned." Heuristics are subjective by nature as general learning points gained through human experience (Giammarco 2012). As Maier and Rechtin state in *The Art of Systems Architecting*, "the format of heuristics is words expressed in the natural languages" (Maier and Rechtin 2009, 31), the 10 heuristics provided in Appendix A are written in the first person and refer to real-life examples and experiences of the author of this report. As such, the writing style used in Appendix A differs drastically from that of the body of this report, in order to communicate the more subjectively derived knowledge often expressed in heuristics and lessons learned.

V. IMPLEMENTATION OF MODEL BASED SYSTEMS ENGINEERING AND ENTERPRISE SYSTEMS ENGINEERING TECHNIQUES AT THE SPACE AND MISSILES SYSTEM CENTER

A. TRANSITIONING TO MBSE

According to Friedenthal, Moore, and Steiner,

> Models and related diagramming techniques have been used as part of the document-based systems engineering approach for many years, and include functional flow diagrams, behavior diagrams, schematic block diagrams, N2 charts, performance simulations, and reliability models, to name a few. However, the use of models has generally been limited in scope to support specific types of analysis or selected aspects of system design. The individual models have not been integrated into a coherent model of the overall system, and the modeling activities have not been integrated into the systems engineering process. The transition from document-based systems engineering to MBSE is a shift in emphasis from controlling the documentation about the system to controlling the model of the system. MBSE integrates system requirements, design, analysis, and verification models to address multiple aspects of the system in a cohesive manner, rather than a disparate collection of individual models. (Friedenthal, Moore and Steiner 2012a, 20)

As Friedenthal, Moore, and Steiner suggest, it is not trivial for an organization to adopt a fully integrated MBSE approach. Furthermore, it can be extremely challenging for an organization to adopt a principle such as MBSE using a common language and approach, such as SysML. The DoD is certainly not immune to these challenges. In fact, the DoD may be an organization where implementing such a transformative change would be the most challenging. Therefore, a gradual implementation of the MBSE principles and use of the SysML language to realize the full potential of MBSE is proposed. Since attempting to adopt both MBSE and SysML at once would likely prove extremely difficult, if not impossible, options are explored for first adopting the principles of MBSE without a wholesale change out of existing DoD processes and practices.

One approach to more gradually implement MBSE principles and practices across an organization such as the DoD is to implement a data exchange specification which

integrates the tools, techniques, and processes currently used by the organization. Using a data exchange specification would allow an organization to continue operating using its currently defined processes and simultaneously realize significant benefits from the ability to integrate and relate these processes and products across organizational boundaries which might have previously been stove-piped and closed off to each other. In highlighting interoperability, the *OMG SysML Version 1.3* specification introduces one such data exchange specification—the ISO 10303 Application Protocol 233, or AP233. (Object Management Group 2012, 8)

B. DATA EXCHANGE SPECIFICATIONS

Data exchange specifications, when applied as a uniform standard across an organization, can bring about significant improvements across an enterprise without forcing drastic change on the organization's current operational rhythm. Any data exchange specification that might emerge as the industry standard, so long as it is mandated as a standard across the organization in question, can effectively realize these benefits. In order to remain consistent with the current specification under development in conjunction with SysML, this report will focus on one particular data exchange standard for description purposes, the ISO 10303 AP233 data exchange specification.

According to the Object Management Group, AP233 is an industry standard metadata model to enable the sharing and exchange of data across multiple tools and across multiple acquisition programs. (SysML and AP233 Mapping Activity 2010) It describes the theoretical or potential linkages between different systems engineering (SE) tools and products and aims to develop a common schema for storing the language of each of these SE tools. This common schema would then enable the export and import of data between compatible SE applications and platforms to better integrate the models and products generated by the applications.

While the AP233 initiative aims to develop an industry-wide metadata standard for all systems engineering, design, and product life cycle management (PLCM) tools, the concept of communicating data between SE platforms is not new. Efforts have been made to link project management and systems engineering tools, such as integrating data

between Microsoft Project and the CORE Systems Engineering tool developed by the Vitech Corporation (Bruring 2009). Enterprise Architect, another well-known SE toolset, has a "Model Driven Architecture (MDA)" initiative aimed at similar objectives to those of AP233. MDA describes translating data from Platform Specific Models (PSMs) into a Program Independent Model (PIM), which can then be imported to other PSMs to share data between platforms (Object Management Group, Sparx Systems 2007).

AP233 divides the tools it prescribes for integration into two categories: Program Management and System Requirements/Design. It also describes the capabilities often used to bridge the gap between these two categories—risk analysis, issue resolution, and management authorization and review—and proposes using the AP233 information model to facilitate the exchange of information through these capabilities. Figure 52 summarizes some of these AP233 tools and highlights the relationships between the AP233 data and the core products of SysML (SysML and AP233 Mapping Activity 2010). Note the similarity of these categories to the technical management and technical systems engineering processes identified by DOD Instruction 5000.02 and summarized in Table 5 and repeated (Figure 52).

Technical Management Processes	Technical Processes
Technical Planning	Stakeholder Requirements Definition
Decision Analysis	Requirements Analysis
Technical Assessment	Architecture Design
Requirements Management	Implementation
Risk Management	Integration
Configuration Management	Verification
Technical Data Management	Validation
Interface Management	Transition

Table 5. Department of Defense—Systems Engineering Processes (From Defense Acquisition University 2013i)

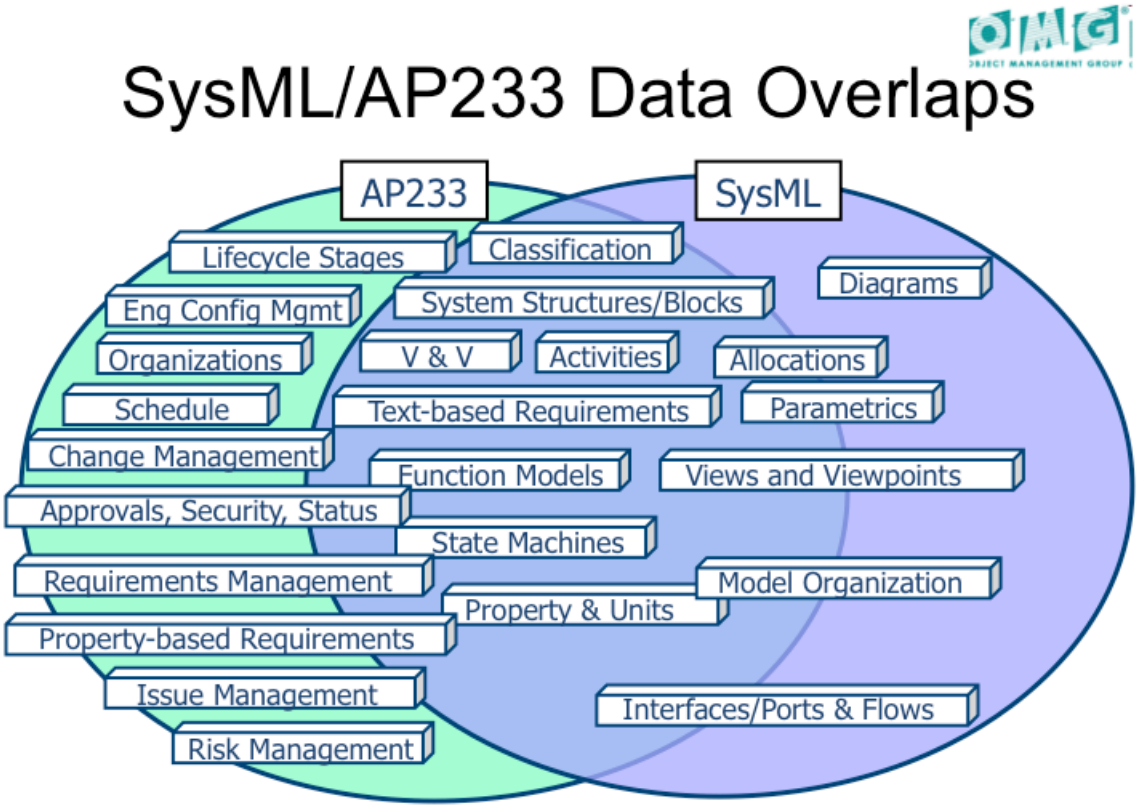

Figure 52. SysML and AP233 Data Overlaps (From "SysML and Ap233 Mapping Activity," OMG SysML Portal Website 2010)

By facilitating the transfer of data and information between multiple program management systems, engineering applications, and platforms, a data exchange specification such as AP233 can enable large organizations and their partners to realize the inter-model relationship benefits described by model based systems engineering (U'Ren 2003).

C. SMC REQUIREMENTS AND CURRENT TOOLS

1. Current SMC Tools and Processes

The great complexity of space systems acquisition requires the use of various design and management tools to adequately characterize and track the system as it progresses through the acquisition lifecycle. The Space and Missile Systems Center (SMC) uses many such tools to generate models and reports intended to help characterize

the system and support decision making. The value of these tools, however, is limited by the fact that they exist at varying levels in the program, from center-wide and program specific tools within SMC to the unique tools and processes used by the prime and subcontractors for each specific program.

Currently, SMC program offices struggle to manage various aspects of system development in a unified way such as described by MBSE. Instead, they are left with an inconsistent and fragmented set of systems, tools, and products that lack clear structure, direction, and governance. Each individual or program element is left with the challenge of identifying which combination of tools, resources, and systems must be used to accomplish their jobs. This current approach lacks guidance as to which resources are available and how they might impact the individual's work, and in many cases, this approach does not provide access to these resources as required. A need exists to provide this guidance and access and to integrate the products so they can be distributed and effectively utilized throughout the program office.

Many of the tools used throughout SMC are not integrated with other tools and products, which oftentimes results in significant disconnects, re-work, and inconsistencies between program elements. This is particularly an issue with system design and architecture tools intended to support concept and system definition, interoperability and interface analysis, and verification and validation activities. Each program has many different methods and tools by which individuals (mainly the contractors) manage these types of design models, and the SMC product center as a whole has very little understanding of what this tool set is.

Tools used to track and manage many of the program management functions—such as requirements, organizational structures, schedules, budgets, and project management status—seem to be more standardized and integrated within SMC than the system design tools. For example, the Comprehensive Cost and Requirement System (CCaR) is used by every SMC program office to capture and track requirements and accounting data. CCaR correlates and traces the system requirements to the project Work Breakdown Structure (WBS) and budget obligations/expenditures. The System Metric and Reporting Tool (SMART) is used by every SMC program office to report program

status information to key stakeholders on a monthly basis. The inputs entered into SMART and CCaR are later integrated and presented in a new application called Executive CCaR which is used by senior DoD acquisition executives to gain insight into their program portfolios. Although these tools are fairly well standardized throughout all of the program offices at SMC, the input data must be manually generated for them, as it is not made readily available from other tools.

Much of the requirements, accounting, and program status data captured by tools like CCaR and SMART is derived from a wide variety of non-standardized and inconsistent applications, tools, and processes. This data, in many cases, is ultimately derived from issue tracking, risk management, and system design and architecture tools. It is within these tools that significant inconsistencies exist between, and even within, the many space acquisition programs at SMC. Furthermore, there are a great number of unique tools and processes used to track issues and manage risks within each program office, resulting in confusion between the various program elements. This confusion is further elevated when the issues and risks being tracked are translated from the tools used by the SMC program office to those used by the prime and subcontractors. The architecture and design for any particular system is generally captured and managed by the contractor's tools and processes, which are even less standardized between the prime and subcontractors and the various SMC program offices.

2. SMC Requirements

In order to fully understand a system's architecture, make well-informed decisions, and report accurate information about a space acquisition program, SMC requires insight into how issues and risks are tracked and how requirements and associated design issues are traced throughout all levels of the program. System models play a key role in supporting SMC program management, such as those models documenting the system's design and build architectures. The program office needs to ensure that requirements are being addressed properly from the mission capability requirements all the way down to the lowest subcontractor level. Responsibility for these requirements must be derived, assigned and tracked in such a way that the system can be

integrated to ultimately fulfill its intended mission. When system issues are discovered from tests or other means, SMC needs to ensure that these issues are properly assigned to the appropriate program elements. Management requires knowledge into where and how these issues are resolved, along with details on the impact of these issues on the overall program. Each program office requires tools and processes that facilitate configuration management of the many documents, models, and products generated and used throughout the organization. To ensure that the system will ultimately meet its requirements and fulfill its mission, SMC needs to understand the system architecture and how the requirements and issues are traced to the components within the architecture. To assess the quality of the system under development, SMC requires the ability to see into both the "as designed" and "as built" architectures.

In order to adequately support each SMC program office and share insight and lessons learned between the program offices, the SMC staff directorates require real-time access to the many program management and systems engineering products generated by the program offices. In addition to simply having access to the program office information, these staff directorates must also interpret and communicate this information to other program offices and to agencies outside of SMC. In order to better understand and communicate this information, these staff directorates require a standardization of the information, documentation, and models generated by each of the program offices.

The current environment makes it very difficult, and oftentimes impossible, for SMC to have the level of visibility into the requirements, issues, risks, and architecture products required to adequately assess and make decisions in support of the program. A basic structure exists to report the status of a major space acquisition program (e.g., CCaR, SMART, Executive CCaR), but the structures and tools used to obtain and track the information necessary to input into this reporting structure—and ultimately ensure that the system's mission will be achieved—is inconsistent and lacking.

D. POTENTIAL VALUE OF MBSE AND DATA EXCHANGE SPECIFICATION TO SMC

The Space Acquisition infrastructure spans a wide range of communities throughout and external to the DoD, including but not limited to acquirers, contractors, concept developers, sustainers, politicians, special interest groups, and of course, users. Each of these communities and stakeholders has different priorities and expectations of a system, and each requires unique insight into the systems' acquisition lifecycle. Being able to communicate system characteristics between these various stakeholders is critical to the success of a program and is an area that the proper application of MBSE principles can support by facilitating different viewpoints. Facilitating different viewpoints, however, is extremely difficult if the processes of each stakeholder are not in sync and the data behind the products are not fully integrated within a common language set. Here is where a data exchange specification comes into play, enabling the translation and integration of the data supporting different viewpoints based on a common model. By endorsing an MBSE strategy with compliance to a data exchange specification such as AP233, SMC could take the first step toward enabling greater interoperability among the various stakeholders. As previously discussed, SysML was designed with AP233 interoperability in mind, and therefore is postulated as the next logical evolutionary step for an organization to realize the maximum benefits of MBSE.

A data exchange specification would facilitate the export and import of data between compatible management and engineering applications and platforms used by all of the key SMC stakeholders, with migration towards a shared information space that can be used to support programmatic decision making throughout the system's life cycle.

Early in the acquisition lifecycle before a SMC program office is formally established, on-going concurrent efforts by the user and concept development organizations are in work to identify capability needs and mission requirements and to study potential material solutions to meet capability gaps. As AFSPC conducts capability based assessments (CBAs) to assess the user's needs, they generate a series of descriptions and products (such as the ICD) identifying current capability gaps and mission-level requirements and measures. Simultaneously, the concept development

organizations are exploring the realm of the possible in efforts to define methods and/or material solutions to fill the capability gaps and meet the requirements. Currently, the various stakeholders participating in these efforts generate these products and capture this information using unique models and views—through unique and disjointed processes, systems, and tools. Applying MBSE principles to standardize the models and views used to capture this common information could result in a synergistic effect and significantly improve the value and quality of the information, models, and products generated. Furthermore, introduction of a data exchange specification could provide the framework necessary to integrate these products across organizational boundaries, enabling all participating stakeholders to better communicate and leverage each other's information and concepts. This framework could then support more commonality across the various stakeholders, such as adoption of SysML.

Once a capability gap is identified and translated to mission-level requirements, and a concept for a material solution is selected, the process of standing up a SMC program office and formulating an acquisition program/strategy is initiated. Although the details and composition of the program office are new at this stage in the lifecycle, the information basis associated with the requirements and the system concept should be well defined and available. In order to avoid duplication of effort and to maximize efficiency, the same standard models and views defined by MBSE principles and used by AFSPC and the developmental organizations to capture these requirements and mission concepts should be flowed to the emerging SMC program office. Again, this flow of MBSE standardized models, views, and system definitions would be made possible through application of a data exchange specification.

The value-added to a program office (and ultimately the Space Acquisition Enterprise) through the use of MBSE principles and data exchange specification concepts as described above increases exponentially as the system descriptions, architectures, models, and views are further flowed down and connected to other key stakeholders, such as the prime and sub-contractors, maintainers, and ultimately back to the operators and users. Furthermore, by expanding MBSE and the utilization of a data exchange specification across the Space Acquisition Enterprise and for each SMC program office,

other SMC staff organizations such as the Program Management and Integration Directorate and the Engineering and Architectures Directorate could provide significantly improved cross-program support with the ability to compare products and correlate heuristics and lessons learned between programs. In addition, these staff organizations could more effectively communicate and support issues both between program offices and to other stakeholders external to SMC such as Congress, AFSPC, and other DoD product centers.

Through the proper application of model based systems engineering principles, in conjunction with a data exchange specification, SMC and the Space Acquisition Enterprise could realize significant improvements to its current processes. Models and products of specific interest to and generated by each key stakeholder could be developed and managed within their unique set of program management and systems engineering applications and processes, improving the flexibility for the organization to meet their specific needs while also leveraging the data, models, and products generated by other key stakeholders. This integration of products and processes would set SMC (and the DoD Space Acquisition community) on a path to alleviate the programmatic issues currently plaguing the Space Acquisition Enterprise as a result of the inconsistent and fragmented systems, tools, and products.

E. BARRIERS AND LIMITATIONS

Taking even this first step will not be easy. There are many potential barriers and limitations—technical and non-technical—to implementing an enterprise-wide model based systems engineering and data exchange specification initiative across the Space and Missile Systems Center and likewise other DoD product centers.

Implementation of a data exchange specification is subject to many technical challenges. In particular, any data exchange specification must interface with and integrate many legacy systems and tools and is therefore subject to the governing rules, restrictions, and shortcomings of each of these systems and tools. Each application has its own unique specifications, routines, data structures, object classes, and protocols that it uses to communicate within itself and between other applications. The introduction of

data exchange protocols could be constrained by some of these unique application properties, and even has the potential to negatively impact these properties and threaten existing application functions. As it is integrated into these applications, systems, and tools, a compliant model must also be implemented within the rules and structure of the physical network and server environments of all key stakeholders, including military organizations, contractors and external partners. Firewalls and other network configuration settings currently restrict the flow of data between SMC and other organizations and corporations such as the prime and subcontractors. Currently, these firewall and network configuration settings even restrict the flow of some information between SMC and its Federally Funded Research and Development Center (FFRDC)—The Aerospace Corporation. A data exchange specification concept will also be subject to a series of Air Force, DoD, and federal instructions, laws, policies, and directives guiding and constraining its use. Table 6 lists some of the most critical of these governing directives.

AF/DoD/Federal Governing Directives
DOD Directive 8500.01 Information Assurance
DOD Directive 8300.02 Data Sharing in a Net-Centric Department of Defense
DOD Instruction 8520.2 Public Key Infrastructure (PKI) and Public Key (PK) Enabling
DOD Instruction 8500.2 Information Assurance (IA) Implementation
Federal Service Oriented Architecture (2008)
DOD Enterprise Services Designation (2009)
Air Force Enterprise Information Management CDD (2003)
Air Force Instruction 33-103: Requirements Development and Processing
Air Force Instruction 33-332: Privacy Act Information
DoD O-5200.1-I (Classified Publication): Index of Security Classification Guides (U)
Federal Information Systems Management Act (2002)
Federal Records Management Act
The Clinger-Cohen Act
Federal Acquisition Regulation (DFAR Sup)
Executive Order 13011, Federal Information Technology
OMB Circular A-130
Federal Enterprise (Information) Architecture Framework (1999)
Air Force Information and Data Management Strategy Policy (2004)
AF-CIO Policy Memorandum 04-12 Mandatory Use of Air Force Enterprise
Information Management Tool Suite (2004)
OSAF-XC Memo - Enterprise Information Management Tool Suite (2006)
AFSPC EIM Strategy (2008)

Table 6. Summary of Key Air Force, DoD, and Federal Information Technology Governing Directives

In addition to the potential technical barriers and limitations facing the implementation of a data exchange specification, there are several non-technical concerns with implementing the model based systems engineering, data exchange, and SysML concepts that must be considered. The concept of working to common models and views as described by MBSE could meet cultural resistance within SMC, as it would involve a significant change in the way the product center currently does business. There could also be contractual limitations associated with the sharing of information between the various program offices and their individual contractors and external partners. The security and/or the proprietary nature of the information being exchanged between various applications, and the classification level of the resultant aggregate information presented by the application, could also be a concern from the government's and contractor's perspective.

Each organization has its own unique governance, policy, doctrine, and business processes which must be adhered to. Introducing MBSE concepts, along with a data exchange specification and/or the SysML language to existing processes and applications could have an impact on how the using organization manages the tools, and ultimately affect how decision makers within the organization interpret products generated by the tools. By making some of the changes necessary to accommodate the introduction of MBSE, other related processes or policies could suffer, and in some cases, these necessary changes could be constrained completely by higher level policies. For example, mandating a tool compliant with a specific data exchange specification within SMC may not be consistent with existing coordination efforts outside of SMC; an Air Force Space Command or DoD policy could require that the same information be reported to them in a different format and through a different process. As a result, the ability of SMC to modify its processes and manage the system acquisition through common models as described by MBSE (and compliant with a data exchange specification and SysML) may be constrained by processes at a higher organizational level. This example serves to highlight the fact that a data exchange specification can only effectively enable the application of MBSE practices, and SysML can only fully optimize the MBSE practices, if all organizations and key stakeholders involved uniformly adopt the standard and adjust their policies accordingly.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSIONS AND RECOMMENDATIONS

A. RESPONSE TO RESEARCH QUESTIONS

The objective of this thesis was to explore the potential benefits of a MBSE approach, in this case, using a structured architecture modeling language such as SysML to develop and employ mission area architectures to DoD space systems. While detailed answers to the specific research questions this thesis sought to answer (introduced in Chapter I) have been discussed and provided throughout the body of this thesis, a summary of answers to the research questions is provided here.

1. What Methods, Techniques, and Processes can be Employed to Aid in the Development of Mission Area Architectures for Department of Defense (DoD) Space Systems?

MBSE is a formalized approach to modeling and architecting a system that can be used to represent all aspects of a system across the full system lifecycle. The Department of Defense Architecture Framework provides the baseline structure and common data meta-model specifications to develop mission area architectures for the DoD, including space systems. Data exchange specifications, such as AP233, can be implemented across a DoD organization to standardize the exchange of architecture and system data between otherwise stove-piped organizational components, enabling synergistic benefits to data analysis across the enterprise. Furthermore, there exist many structured techniques, applications, and languages to enable and aid in the development and assessment of detailed space system architecture, capturing the detailed interactions and interdependencies within and throughout a system and enabling rigorous mathematical analysis to support key programmatic decisions and needs, including the Structured Analysis and Design Technique and the SysML.

2. In What Ways or in What Instances Can Model Based Systems Engineering (MBSE) be Used in the Development of Space Based Mission Area Architectures for the DoD?

The principles of model based systems engineering provide the framework for organizations to select a set of interrelated models to help characterize and analyze a

system and document the design, acquisition and sustainment process. Data exchange specifications, such as AP233, can be mandated across the DoD to enable MBSE practices and benefits across the enterprise. However, in order to realize the maximum benefits of MBSE, including enhanced communications, reduced development risk, improved quality, increased productivity, and enhanced knowledge transfer, a structured architecture development technique such as the Structured Analysis and Design Technique or SysML must be implemented across the DoD space community and used to develop space based mission area architectures. Specific examples of MBSE principles applied to the Overhead Persistent Infrared mission area architecture are provided within the case study in Chapter IV of this thesis.

3. How can the System Modeling Language (SysML), Based on the Common Software Engineering Unified Modeling Language (UML), be Applied to Aid in Developing Mission Area Architectures for DoD Space Systems?

A detailed example of the application of SysML, in conjunction with MBSE principles, is provided for the OPIR mission area, and specifically modeled for the SBIRS system. This executable SysML model, once complete and specified with mathematical relationships, can be used to support rigorous engineering analysis. SysML is consistent with the DoDAF specification and supports complete end-to-end realization of MBSE practices and benefits throughout the acquisition lifecycle. Ultimately, the overall quality of a system acquisition effort, or project, can be greatly improved through the structured application of MBSE architecture, modeling and simulation, and trade study activities—all enabled by the development of architecture using SysML.

B. PROCESS DISCUSSION

1. Discussion of the Iterative and Recursive Nature of the Synthesis Process

Throughout the process of designing the SBIRS case study architecture presented in this paper, many issues were encountered and many design decisions were re-evaluated. After decomposing the system to the first level functional architecture, the author of this thesis looked back at the external systems diagram and made design

modifications to the inputs and outputs between the system, the user, and all external information systems. As the decomposition and I/O of the functional architecture evolved through multiple design iterations, so did the physical architecture and its allocation to the system functions. Significant changes were also made to the allocation of the physical components to the functional architecture during the definition of the interfaces and links within the model. There were even minor design changes being made during the definition and allocation of input/output and non-functional requirements to the SBIRS design architecture. As this reiteration played out continuously throughout the development of the SBIRS architecture, changes would not simply be limited to the design architecture. Other iterative design modifications during and throughout the synthesis process might include changes to the SBIRS concept of operations, including modifications to the context diagram and use case scenarios to more accurately reflect the type of information being exchanged between the key system stakeholders and external systems.

2. Comments on the Use of MagicDraw

The use of a software tool such as MagicDraw is necessary, in this researcher's opinion, for conceptualizing and architecting a complex system such as SBIRS. The key aspect of the software that makes it so critical in the design and architecture activities is its affiliation with a central database of elements and their relationships. Since the design of a complex system is very iterative and recursive, changes to the functions, components, interfaces, links, inputs, outputs, controls, mechanisms, and many other model relationships can be expected throughout the design process. Because of these frequent changes, having the properties and relationships of each element centrally stored in a database becomes critical to maintaining the integrity of the design, not to mention tremendous savings in time and frustration.

Because of this database centric capability, an architecture and design tool such as MagicDraw further shows its value by providing the designer with guidance in modeling syntax and allowable relationships while providing the capability to generate a set of consistent standardized models and diagrams from information provided to the same

central database. This promotes common modeling syntax standards and patterns to aid in communication of a design between multiple parties and disciplines—design communication being a principle purpose of developing these models in the first place.

C. CONCLUSIONS

MBSE, in conjunction with SysML, can provide extremely powerful benefits to the development of architectures and across the acquisition life cycle of DoD space systems.

> MBSE can provide additional rigor in the specification and design process when implemented using appropriate methods and tools. However, this rigor does not come without a price. Clearly, transitioning to MBSE underscores the need for up-front investment in processes, methods, tools, and training. It is expected that during the transition, MBSE will be performed in combination with document-based approaches. For example, the upgrade of a large, complex legacy system still relies heavily on the legacy documentation, and only parts of the system may be modeled. Careful tailoring of the approach and scoping of the modeling effort is essential to meet the needs of a particular project. (Friedenthal, Moore and Steiner 2012a, 20–21)

Adopting MBSE and SysML for the design of DoD space systems will require a fundamental paradigm shift in how the DoD does business, transitioning from what is now a purely document-driven approach. The implementation of a standard data exchange specification can be applied to realize some enterprise benefits and aid in the development of requirements for a more integrated and rigorous approach, such as SysML. It is clear that such a paradigm shift is required if the DoD and its space acquisition element, SMC, is to meet its requirements and realize the powerful benefits promised by MBSE and SysML.

D. RECOMMENDATIONS

As discussed, significant value can be added to SMC and the space acquisition enterprise in response to its requirements by adopting and pursuing a robust model based systems engineering structure supplemented and enabled by endorsing a data exchange specification such as the AP233 metadata model standard as a first step to full adoption of MBSE and SysML. Although many potential barriers and limitations exist that may

limit or impede the introduction of MBSE practices and the application of a data exchange specification, it is recommended that SMC first moves towards the MBSE and standard data exchange initiatives to realize both short and long term improvements in its processes and decision making ability in support of the acquisition of DoD space systems. As space systems become more and more complex, having robust processes and clear system characterizations in place, as described by MBSE, will become increasingly necessary to successfully design and acquire a space system. In order to minimize the impact of the implementation barriers described, a phased approach is proposed.

1. Phase 1

Since many of the issues currently plaguing SMC are related to the inconsistent and fragmented set of tools and processes used within and between the various program offices, the first incremental phase towards improving the efficiency of SMC's acquisition processes is to identify, list, and manage the configuration of all critical program models, processes, and tools used throughout the center. This short term goal will not only help SMC better understand where and how the program management and systems engineering functions are executed and tracked by the various program offices, but it will also identify where common models, processes, and tools can be adopted and standardized in the future. This will be the first step in identifying the set of models and views that will meet the requirements of SMC and shape the eventual MBSE structure.

2. Phase 2

Once the current models and processes have been identified, SMC should integrate these models, processes, and tools across the many program offices, staff directorates, and external partners, including the prime contractors, The Aerospace Corporation, and Air Force Space Command, using the principles of MBSE and the capabilities of a standard data exchange specification, such as AP233. Realizing this will require several incremental milestones and will involve significant communication and coordination between a wide range of different organizations, but if SMC begins to advocate for model based systems engineering and endorses a standard data exchange

specification now, the significant improvements to SMC and the Space Acquisition Enterprise discussed within this paper could eventually become a reality. Furthermore, understanding of the requirements for implementing SysML across the enterprise will be much more complete and clear following the implementation of a standard data exchange specification.

3. Phase 3

While great progress could be made to current processes used in space system acquisitions, simply implementing a data exchange specification would not fundamentally improve how information is managed at the component level. Great strides can be made to improving the enterprise if the community can make the transition from a document-based system, as described earlier and effectively left un-changed with the adoption of just a data exchange specification, to a true model based system as prescribed by MBSE. In order to achieve this, a common language must be adopted across the DoD space acquisition enterprise that focuses on not just assessing but also generating and developing program data and architectures using MBSE tools and techniques.

Given adoption of MBSE practices, and the relaxation of the barriers between the many varied stakeholders of SMC and the larger DoD Space Acquisition Enterprise, the community should consider further standardizing its implementation of MBSE practices by enforcing common processes, standards, models, tools, and techniques across the community. As discussed within this paper, the SysML modeling language is uniquely suited to meet this demand. With the enterprise-wide adoption of MBSE practices and the standard SysML language, the DoD Space Acquisition community could truly realize all of the powerful benefits described within this paper, and ultimately deliver more successful systems through more effective acquisition efforts.

E. FUTURE WORK

The SysML architecture model and MBSE practices described herein are ultimately only as useful to an organization as the underlying data that they represent is complete, clear, stable, and consistent. Therefore, further study would assess techniques

and methods for assessing the quality of the SysML architecture itself and the relative maturity of the MBSE products and practices. The applicability of architecture assessment methods, such as those introduced by Kristin Giammarco in "Formal Methods for Architecture Model Assessment in Systems Engineering" (2010) and "Architecture Model Based Interoperability Assessment" (Giammarco, Architecture Model Based Interoperability Assessment 2012) to the MBSE and SysML architecture techniques described in this report could be further studied. The system architecture heuristics provided in the Appendix could then be quantified and applied precisely to assess the quality of the architecture and models developed using SysML and MBSE.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX. THE ART OF SYSTEM ARCHITECTURE—HEURISTICS

As "the format of heuristics is words expressed in the natural languages" (Maier and Rechtin 2009, 31) the 10 heuristics provided in this appendix are written in the first person and refer to real-life examples and experiences of the author of this report. As such, the writing style used differs drastically from that of the body of this report, as is suitable for use when discussing heuristics

1. Focus on User Interactions and Interfaces

A system is not likely to be accepted by a user if the user's interaction with the system is not intuitive and what they would expect from a similar system. When architecting, pay particular attention to how the user will interact and interface with the system. Imagine how the system would "feel" in the hands of a user, put yourself in the user's shoes, and ask yourself how you would expect to interact with the system.

a. Discussion

I am a longtime fan of Nintendo and have owned nearly every gaming system they have produced over the years. Of these systems, I have always been particularly interested in their portable gaming platforms and have owned everything from a classic Game Boy to a Nintendo DS. I recently purchased the Nintendo DSi XL. Immediately after removing the product from the box, I was struck by this architecture principle.

All DS systems have two display screens that open about a central hinge holding the two together. When I opened my new Nintendo DSi XL, I immediately noticed that the hinge holding the two screens together was loose, causing the system to wobble during play. While all previous versions of the DS system (DS, DS Lite, and the DSi) had the same basic screen design, the DSi XL is the first to exhibit this wobble at the hinge. As alluded by the "XL" in its name, the primary difference between the DSi XL and its predecessor, the DSi, is the increased size of the gaming system and size of the display screens (an increase of 93 percent). However, while they increased the size of

the display screens by 93 percent, Nintendo did not redesign the hinge to account for the additional weight of the top screen caused by its larger size.

Had the system architect focused more on the components with which the user was interacting directly, he or she may have noted the connection between the screen size and the hinge design. While all other aspects of the new system were clearly an improvement from its predecessor, the DSi, the wobble of the screen became quickly apparent and was not what I expected to "feel" when using the system. My acceptance of the system was therefore threatened by the uncharacteristic and unexpected behavior of a critical system component that I came in direct contact with each time I used the system.

I observe this very same principle daily as I surf the Web. After accessing common web services, such as those used for banking, searching, shopping, as well as online encyclopedias, I come to expect a particular "look and feel" when accessing other similar Web services. If the user interface's design of these services is radically different than others, I have a tendency to quickly reject that system or service as a result.

2. Maximize Cohesion

Assign specific responsibilities and scope to the system elements, and stick to them! Individual aspects of a system architecture must be clearly focused to ensure that they are understandable, manageable, and supportive of an open and robust system design. Highly cohesive architecture elements are understandable and manageable because they comprise similar functions, activities, or operations and "bucket" them logically. They support an open and robust system design because they can be complemented by other cohesive elements to meet future requirements by reducing the number of interfaces and supporting low coupling within the system. When developing system architecture, clearly define the purpose and scope of each element of the architecture. When data items, functions, activities, or other elements need to be added to the system, assign those elements to a component that is closely related to that element or that serves a similar purpose. If the new element does not easily fit within a currently defined "bucket," consider creating a new "bucket" or component to host that element. Avoid the tendency to group activities, functions, data items, or other elements that are

not closely related as this has the potential to increase the coupling and therefore the complexity of the system. This increased complexity will likely result in increased cost, schedule, and performance risk. If a new capability is desired and does not fit the initial scope, do not attempt to significantly modify or expand an existing element once the system has been implemented based on highly cohesive elements (Larman 2005, 314–317). This also related to the heuristic: "Group elements that are strongly related to each other, separate elements that are unrelated" (Maier and Rechtin 2009, 402).

a. Discussion

About a year ago, when I first arrived at my current organization, I noticed a shortfall in the way my organization was managing and promoting training and education opportunities to members of the organization. As a result, I talked to the training managers to derive requirements for a system that could help fill this gap. After talking to the training managers and several other stakeholders, including other members of the organization and the organization's leadership, I determined that I would construct a capability on the organization's intranet site for the training managers to post information about upcoming training and education opportunities and advertise these to the organization. The initial scope of this application was as an information and advertising capability only.

Eventually, this capability was adapted by the organization and was such a success that recently an additional requirement to add a registration capability to the training list was proposed. The initial development effort to accommodate this new requirement encountered significant challenges since registering for a course proved to be quite different in functionality than simply listing and advertising the courses. It was quickly realized that trying to modify and add onto the existing training list capability would not be possible without negatively impacting aspects of the list that met its original requirements and scope. Trying to incorporate both of these functions in a single element did not support high cohesion. As a result, the design was not manageable or open/robust to future design.

3. Minimize Coupling

Minimize the dependency of one system component on other system components. The more dependencies that exist between one system, activity, function, or data object and another, the more complex the system becomes and the more likely it is to fail. If one element relies heavily on another element and the configuration of that element changes, there will likely be an impact to the dependent element. This dependency makes the system not-conducive to change. Conversely, a system having elements with low coupling supports an open and robust system design by means of reducing the number of interfaces and therefore the complexity of the system. When architecting a system, be sure to adhere also to Principle 2: High Cohesion. Highly cohesive elements should, by definition, enforce low coupling between elements. Define activities, functions, data items, systems, and other elements such that they depend on other elements as little as possible. If changes are required of the system to satisfy a future requirement, follow this same principle to ensure that changes to one element have a minimal impact on other system elements. Pay particular attention to any dependencies between elements as system modifications are made and ensure that these dependencies are not adversely affected by the changes (Larman 2005, 299–302). This also relates to the heuristic: "Choose a configuration with minimal communications between the subsystems" (Maier and Rechtin 2009, 402).

a. Discussion

Returning to my compulsive tendency to dissect and criticize systems and analyze their designs, I recently purchased a digital wrist watch to time my 1.5 mile run in preparation for my Air Force physical fitness test. I couldn't help but notice a violation of this principle in the software design of the watch. The stopwatch function of the watch has a split display—one half showing the actual timer used while running and the other displaying the function in question. Before starting the timer in the stopwatch function, the top display shows the current time. This is useful for connecting back to the real world if, for instance, a meeting is approaching and I need to ensure that I stop exercising by a certain time in order to make the meeting. The stopwatch also uses the

top display for another function – the ability to set an alarm when a specified time is reached by the timer. This is useful for providing feedback on meeting my run time goals.

Although both functions of the top display are useful, they are coupled in such a way that they become practically useless. This is because the first function—displaying the current time—only shows prior to starting the timer while the second function—displaying the alarm time—only shows while the timer is running. This is exactly opposite of what would be expected since the user is concerned with the real world time while running and sets the alarm time prior to starting the run. The way these two functions are coupled with the stopwatch timer function results in greater system complexity and, in this case, makes two otherwise very useful functions practically useless.

4. Don't Forget Implementation Planning

You can design the best system possible, but if it is not implemented properly, it could still fail. Even a "perfect" system can fail if not implemented at the proper time, in the proper environment, or with the proper configuration and support behind it. Early planning for how a system is to be deployed will shape the entire design effort by defining how to phase the system, set the schedule, and "sell" the system to its key stakeholders.

The moment a system or product is conceived and a concept developed, well before a comprehensive architecture effort, planning the implementation of the system must begin. This planning includes determining when to deploy certain aspects of the system, within which environment to deploy them, how maintenance and upgrades to the system will be conducted, and other business concerns related to the system's design and implementation. This implementation planning must be conducted throughout the lifecycle of the system—"from cradle to grave"—and should be updated and evolved iteratively and recursively with time. This principle is related to the heuristics: "Good products are not enough. Implementations matter" (Morris 1993) and "If social

cooperation is required, the way in which a system is implemented and introduced must be an integral part of its architecture" (Maier and Rechtin 2009, 398).

a. Discussion

I have observed numerous products and systems developed and implemented at Los Angeles Air Force Base that have failed to achieve their full potential due to inadequate implementation planning. Some great systems are designed and developed for use across the LAAFB community only to find that they are not accessible by a group of critical stakeholders who support the project from another organization external to the LAAFB network. This is an example of not implementing the system in the proper environment or not planning for the environment properly.

Another common failure mode of otherwise successful systems is not implementing them at the right time. There are many systems that get locked in development due to requirements creep and other issues such that when they are finally deployed, the environment and user requirements have changed enough to significantly degrade the usefulness of the system or even make it completely obsolete. Early implementation planning for how to phase the deployment of the system could have combated this failure mode by clearly defining an implementation schedule.

Other issues occur when a system is not "sold" or marketed appropriately to its users. Many of the information systems to which I am referring critically rely on a large community using the system to ensure the information is fresh and complete. If the system is not integrated into current business and technical processes, or critical users are simply not made aware of the system's existence, the product is likely to fail as a result.

5. Cannot Optimize for all Stakeholders

You cannot make everyone happy. A complex system cannot equally satisfy and completely meet the needs of all stakeholders. As more requirements are gathered from more and more stakeholders, competing requirements arise for which fully meeting one requirement could result in not meeting another.

Focus on meeting the needs of the key stakeholders first since their acceptance of the system will be critical to success. A "balancing act" must be played to reach a common ground with stakeholder requirements and expectations when these requirements compete with one another. Always be honest with the stakeholders as promises are made to meet specific requirements, particularly when making trade-offs to meet or optimize some requirements at the expense of others. This principle is related to the heuristic: "No complex system can be optimum to all parties concerned, nor all functions optimized" (Maier and Rechtin 2009, 399).

a. Discussion

For complex systems, it is natural for there to be many different stakeholders having very different viewpoints and expectations from the proposed system. I often ask myself, "Why did they design the system this way?" or "Why didn't they put this function in the system?" as well as find myself saying, "This function is useless to me!" Before, I would be relentlessly critical of the systems for which I would make these comments—quickly rejecting the system as a "piece of junk." Now that I am becoming a trained architect and can take a step back to see the system from a higher level, I am more careful not to jump to the conclusion that the system is a "piece of junk" but to first consider the possibility that I am simply not among the stakeholder group on which the system was primarily focused.

Accepting the fact that a system has features that are useless to me or that it does not have features that I believe it should have is difficult to do; however, I can now appreciate the fact that the selection of these features was likely directed towards a specific audience and perhaps is not the correct system for me. An example of this idea is the iPod. Apple designs its products, such as the iPod, to have the simplest user interface possible, often at the expense of including useful features up front. When listening to music, I prefer a variety and therefore find myself using the "shuffle" feature frequently. Since this is one of the most useful features to me, it would be ideal if I could easily toggle the shuffle option with one motion such as by flipping a switch on the device itself. While some systems have this option, the iPod does not since having this

button on the device would detract from a more critical requirement for the product—a simple and clean aesthetic appearance. Of course it would be foolish to reject the iPod as a failed system because of this feature, since history shows that Apple has been extremely successful with the iPod system.

6. Diverse Perspectives

Embrace different opinions. It takes a diverse team of individuals to develop the best systems. Throughout the design and development of a product, system, or service, members of a team will have different perspectives and opinions that influence the design of the system. Only when these diverse perspectives are managed properly can the greatest potential system be realized.

When working with a team of individuals on designing, architecting, and fielding a system, it is crucial that a leader be present to encourage the free exchange of diverse ideas and options related to every aspect of the system. Lead a system design by opening the table to this diversity. Ask targeted questions often to facilitate the exchange of these different perspectives. Adhere to this principle throughout the design of the system and success will be within reach. Ignore it, and failure is likely. This principle relates to the heuristic: "If you think your design is perfect, it's only because you haven't shown it to someone else" (Harry Hillaker 1993, quoted in Maier and Rechtin 2009, 405).

a. Discussion

Throughout my career and professional development, and more recently while working on group projects for my 'master's degree program, I have learned that there are always multiple ways to look at and interpret a problem. As a team progresses though the natural team building model (forming, storming, norming, and performing), I become more and more open to the different opinions and perspectives of my teammates. Being more open and embracing these different perspectives has resulted in us more accurately defining the problem, collecting a pool of potential solutions, and evaluating each potential solution, and it has ultimately resulted in a better product than could have been obtained from an individual effort.

Working with and accommodating for these diverse perspectives is rarely easy and must be approached delicately to maintain an effective working relationship between individuals. In early discussions, it is ideal to brainstorm as many different ideas as possible; however, eventually a consensus must be reached and a decision must be made that, by definition, will not fully satisfy the opinions of all parties. These decision trade-offs are made throughout a system's design and are possibly the most critical leadership challenge for managing a program. If managed properly, adhering to this principle can bring about huge dividends for the system in the long run. If not managed properly or if this principle is not adopted, it is unlikely that the optimal system will be achieved.

7. Maximize Alternatives

Once you make a decision and continue the design effort based on that decision, you are likely stuck with it for the life of the system. The more alternatives and options you can come up with, the more likely you are to come up with the best one.

While designing and architecting a system, come up with as many alternative solutions to every problem as possible and as time permits, and hold on to those alternatives until a decision absolutely must be made to move forward. This principle is related to the heuristic: "Build in and maintain options as long as possible in the design and build of complex systems. You will need them. OR… Hang on to the agony of decision as long as possible" (Robert Spinrad 1988, quoted in Maier and Rechtin 2009, 40).

a. Discussion

I tend to be a naturally indecisive person, a characteristic that I've identified to be both strength and weakness in myself. Indecisiveness can be a weakness in a leader if decisions are never made resulting in a lack of guidance and direction, but indecisiveness can also be a strength when it encourages developing a complete understanding of a problem and trying to arrive at the best possible solution to that problem. An effective leader must balance this with cost, schedule, and other

programmatic issues to arrive at as many different alternative solutions as possible and hold onto those alternatives as long as practical.

Holding onto these alternatives and delaying a decision as long as practical (cost and schedule considered) can bring benefits to that decision, since the more defined the system architecture becomes, the more the architect will be able to understand and comprehend the significance and impact of that decision. The bottom line is: the longer you wait to make the decision, the better you understand the problem and impact of the decision; therefore, it follows that a better decision can be made since it is based on better understanding and more information.

Once that decision is made and subsequent decisions are made based on it, it becomes more and more difficult to reverse or change that decision without resulting in significant rework and redesign. Since schedule will always play an important role in a successful system, there is rarely the time or money to go back and change a decision and accomplish this rework; therefore, every decision made in the design and architecture of a system causes exponential residual effects to ripple throughout the system design. It is therefore critical to get those decisions right the first time. Coming up with as many alternatives as possible and delaying the selection from among the alternatives as long as possible increases the chances of making the best possible decisions and therefore the best possible system.

8. Use Prototypes to Refine Requirements

Users do not know what they need until they can put their hands on it. A user's stated requirements might be what they want at the time, but what they really "need" is another matter. Only once users can touch and feel a system and criticize its design do they truly start defining their needs.

As a means of further understanding requirements and refining user needs and expectations, develop early prototypes of the system or elements of the system. Involve the users and other key stakeholders in testing and operating these prototypes and keep an open ear to their comments, concerns, frustrations, and desires. Accurate requirements can be extracted whether the prototypes meet or do not meet the user's needs and

expectations. Involve the users in this way throughout the development of the system to maximize the chances of success. This principle is related to the heuristics: "The phrase, "I hate it," is direction" (Lori I. Gradous 1993, quoted in Maier and Rechtin 2009, 270) and "The most important single element of success is to listen closely to what the customer perceives as his requirements and to have the will and ability to be responsive" (J. E. Steiner 1978, quoted in Maier and Rechtin 2009, 270).

a. Discussion

As I have developed web services and applications for users at Los Angeles Air Force Base, I have found that using rapid prototyping is by far the best and most efficient way to extract user requirements. The users rarely know what they are looking for until they have a basic user interface or picture to look at and poke holes in. They very quickly begin making comments like "you forgot to add this," "this belongs here, not here," and "I don't like the look of that." Whether they know it or not, these comments are very powerful requirements influencing the design of the eventual system. I listen carefully and take notes during these interactions and then return to the drawing board to incorporate changes to the prototype based on these comments. I then return to the user with the updated prototype and repeat the process to further refine the user's requirements and expectations of the system. Once the criticisms thin and the user begins to like the system more and more, the first (or next iterative) version of the system can be fully developed and deployed.

In addition to helping the user define their requirements and the developer meet the true needs of the user, using rapid prototyping in this way also helps to scope the user's expectations of the system to be delivered. He or she will have a better mental picture of what the system will look like and how it will function since he or she was closely involved with its design. The user will also be more likely to accept the system as he or she develops a sense of ownership for it. Since the user was involved throughout the requirements refinement and design process, he or she is more likely to say, "I designed this system" and "This is my system." This distinction is very important for

system acceptance and implementation since without that sense of ownership, the system is likely to fail.

9. Iterative and Recursive

Although a decision made early in the architecture process may seem trivial, the impacts of that decision, made clear further down the road, could change your perspective completely. In order to fully understand and adequately capture a system, the architecture must be developed iteratively and recursively.

When architecting a system, it is important to keep an open mind and remain flexible to evolution and change as the architect's vision of the system becomes more and more clear. At each level of architecture, and while the system is further and further decomposed, the architect must take a step back to re-evaluate and improve the design at higher levels based on the enhanced understanding of the system gathered from diving down into the lower levels.

a. Discussion

Throughout the process of designing the SBIRS architecture, many issues were encountered and many design decisions were re-evaluated. After decomposing the SBIRS system to the first level, I looked back at the external systems diagram and made design modifications to the inputs and outputs between the SBIRS system, the user, and all external information systems. As the decomposition and I/O of the functional architecture evolved through multiple design iterations, so did the physical architecture and its allocation to the system functions. As this reiteration played out continuously throughout the development of the SBIRS architecture, changes were not limited to only the design architecture. Other iterative design modifications made during the synthesis process included changes to the SBIRS concept of operations, including modifications to the use case scenarios to more accurately reflect the type of information being exchanged between the key system stakeholders and external systems.

10. Modular Design

Carefully applied component commonality equals significant lifecycle cost savings. Long-term cost savings and performance benefits can be realized by a system and its related systems through the use of an open, modular design.

When architecting a system, choose components such that they support the principles of high cohesion (Principle 2), low coupling (Principle 3), and maximize commonality so they can be re-used or repurposed within the system being designed as well as in other related systems. This principle is related to the heuristics: "Use open architectures. You will need them once the market starts to respond." and "Relationships among the elements are what give systems their added value" (Maier and Rechtin 2009, 399).

a. Discussion

Great benefits can be realized by families of systems having components with high commonality. These common components support an open, modular design and can be more easily replaced, updated, and re-used. This commonality has the potential to result in significant cost savings for maintaining the system and can greatly reduce the logistics footprint of it and other related systems, resulting in additional cost savings and increased supportability.

An example of this principle is a family of power tools from the same company. I have a power tool kit that includes a drill, circular saw, sander, radial arm saw, and flashlight all from the same company and manufacturer. Although this kit contains five distinctly different systems they all have one thing in common—the rechargeable battery power supply. I can easily remove the power supply from a power tool and plug it into a recharger (another common system) or transfer it to another power tool. Having this common power supply reduced the cost of the power tool kit since I only had to purchase two power supplies (really only one was necessary, but a back-up is always nice to have) for the five power tools rather than one for each. The cost of the system was likely further reduced by minimizing the complexity of designing, manufacturing, packaging, handling, and shipping the power tool kit.

Further benefits can be seen in the maintenance and logistics of the modular family of power tools. If one of the power supplies fails and needs to be replaced, I can simply purchase another power supply. I can avoid untimely and expensive repairs on a specific power tool since the power supply has been de-coupled from the architecture of the tool itself. Also, if I require a power supply with a greater battery life, I can purchase an upgraded power supply, again without changing or impacting the tool itself. The maintenance of each system is therefore greatly simplified as a result of its modular design, resulting in even more cost savings. The logistics footprint of storing and transporting the set of power tools is reduced, since I only have to store and haul two power supplies instead of five or more.

LIST OF REFERENCES

Atego Corporation. 2013.“Artisan Studio.” Atego Website. Accessed September 10, 2013 http://www.atego.com/products/artisan-studio.

Baker, Loyd, and Ann Christian. 2013. “Requirements Development and Management Using Models.” Vitech Corporation. Accessed 10 September, 2013 http://www.vitechcorp.com/resources/technical_papers/200701031634140.baker_christian.pdf.

Baker, Loyd, Paul Clemente, Bob Cohen, Larry Permenter, Byron Purves, and Pete Salmon. 2013. “Foundational Concepts for Model Driven System Design.” Accessed 10 September, 2013 http://www.vitechcorp.com/resources/technical_papers/200701031636590.baker_etal96.pdf.

Bruring, Pieter. 2009. “Linking CORE with MS Project. Aircraft Development and Systems Engineering (ADSE).” Blacksburg, VA: Vitech Corporation.

Defense Acquistion University. 2013a. “3.3: Analysis of Alternatives.” In *Defense Acquisition Guidebook.* Accessed 10 September, 2013 https://acc.dau.mil/CommunityBrowser.aspx?id=488336&lang=en-US.

———. 2013b.“4.3.12: Architecture Design Process.” In *Defense Acquisition Guidebook.*. Accessed 10 September, 2013 https://acc.dau.mil/CommunityBrowser.aspx?id=638341&lang=en-US.

———. 2013c. “4.3.19.1: Modeling and Simulation.” In *Defense Acquisition Guidebook.* Accessed 10 September, 2013 https://acc.dau.mil/CommunityBrowser.aspx?id=638373&lang=en-US.

———. 2013d. “7.2.5: DoD Enterprise Architecture-Related Guidance.” In *Defense Acquisition Guidebook.* Accessed 10 September, 2013 https://acc.dau.mil/CommunityBrowser.aspx?id=511597&lang=en-US.

———. 2013e. “Chapter 4: Systems Engineering Processes.” In *Defense Acquisition Guidebook.* Accessed 10 September, 2013 https://acc.dau.mil/CommunityBrowser.aspx?id=638325.

———. 2013f. “Defense Acquisition Management System: Technical “V” Activities.” In *Defense Acquisition Guidebook.* Accessed 10 September, 2013 https://acc.dau.mil/CommunityBrowser.aspx?id=294550.

———. 2013g. “Enclosure 12: Systems Engineering.” In *Defense Acquisition Guidebook*. Accessed 10 September, 2013 https://acc.dau.mil/CommunityBrowser.aspx?id=332558&lang=en-US.

———. 2013h. “Operation of the Defense Acquisition System (*DoD Instruction 5000.02*).” Accessed 10 September, 2013 https://dap.dau.mil/aphome/das/Pages/Default.aspx .

———. 2013i. “Systems Engineering Process: Technical Management Processes.” In *Defense Acquisition Guidebook*. Accessed 10 September, 2013 https://acc.dau.mil/CommunityBrowser.aspx?id=294549 .

———. 2013j. “SE Activities: Materiel Solution Analysis Phase.” In *Defense Acqusition Guidebook*. Accessed 10 September, 2013 https://acc.dau.mil/CommunityBrowser.aspx?id=294551.

Department of Defense. 2009. *DoD Architecture Framework Version 2.0: Architects Guide*. Department of Defense. Accessed 10 September, 2013 www.prim.osd.mil/Documents/DoDAF_2-0_web.pdf.

Elm, Joseph P. and Dennis R. Goldenson. 2012. *The Business Case for Systems Engineering Study: Results of the Systems Engineering Effectiveness Survey*. Pittsburgh, PA: Carnegie Mellon University.

Friedenthal, Sanford, Alan Moore, and Rick Steiner. 2012a. “Chapter 2: Model-Based Systems Engineering.” In *A Practical Guide to SysML - The Systems Modeling Language*, 15-27. Waltham, MA: Elsevier Inc..

———. 2012b. “Chapter 3: Getting Started with SysML.” In *A Practical Guide to SysML-The Systems Modeling Language*, 29-49. Waltham, MA: Elsevier Inc..

———. 2012c. “Chapter 8: Modeling Constraints with Parametrics.” In *A Practical Guide to SysML-The Systems Modeling Language*, 185-204. Waltham, MA: Elsevier Inc..

Fusion Forge. 2013. “TOPCASED-SYSML.” Fusion Forge. Accessed 10 September, 2013 http://gforge.enseeiht.fr/projects/topcased-sysml/.

Gau Pagnanelli, Christi A., Barbara J. Sheeley, and Ronald S. Carson. 2012. “Model-Based Systems Engineering in an Integrated Environment.” 22nd Annual INCOSE International Symposium. Rome, Italy, July 2012..

Giammarco, Kristin. 2012. “Architecture Model Based Interoperability Assessment.” PhD diss., Naval Postgraduate School, Monterey, CA.

———. 2010. “Formal Methods for Architecture Model Asssessment in Systems Engineering.” Paper presented at the 8th Annual Conference on Systems Engineering Research. Hoboken, NJ.

International Business Machines (IBM). 2013.“Rational Tau.” IBM. Accessed 10 September, 2013 http://www-03.ibm.com/software/products/us/en/ratitau/.

———. 2013. “Rational Rhapsody Family.” IBM. Accessed 10 September, 2013 http://www-03.ibm.com/software/products/us/en/ratirhapfami.

International Council of Systems Engineering (INCOSE). 2011. “4.3: Architectural Design Process.” In *INCOSE Systems Engineering Handbook v. 3.2.2*, 96-115. San Diego, CA: International Council of Systems Engineering, 2011.

———. 2007. “Models and Model-based Systems Engineering.” In *Systems Engineering Vision 2020*, 15. Technical Operations, INCOSE.

———. 2004. “What is Systems Engineering?”. Accessed 10 September, 2013 http://www.incose.org/practice/whatissystemseng.aspx.

Los Angeles Air Force Base. 2013. “Infrared Space Systems Directorate.” Los Angeles Air Force Base. Accessed 10 September, 2013 http://www.losangeles.af.mil/library/factsheets/factsheet.asp?id=5330.

Larman, Craig. 2005. “Chapter 17: GRASP: Designing Objects with Responsibilities.” In *Applying UML and Patterns - An Introduction to Object-Oriented Analysis and Design and Iterative Development*, 271-319. Upper Saddle River, NJ: Prentice Hall.

Liston, Paul, Kamil Erkan Kabak, Peter Dungan, James Byrne, Paul Young, and Cathal Heavey. 2010. “Chapter 11: An Evaluation of SysML to Support Simulation Modeling.” In *Conceptual Modeling for Discrete-Event Simulation*, by Stewart Robinson, Roger Brooks, Kathy Kotiadis and Durk-Jouke van der Zee, 279-307. CRC Press..

Long, Dave. 2010. “Lecture 7: Process/Activity Modeling.” Lecture, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH.

Long, David and Scott Zane. 2011a. “Language: The Systems Model Is Language-Based.” In *A Primer For Model-Based Systems Engineering* (2nd ed.), 37. Blacksburg, VA: Vitech Corporation.

———. 2011b. “Characteristics of a Model.” In *A Primer For Model-Based Systems Engineering* (2nd ed.), 33. Blacksburg, VA: Vitech Corporation.

———. 2011c. “Four Elements of a Model.” In *A Primer For Model-Based Systems Engineering* (2nd ed.), 32. Blacksburg, VA: Vitech Corporation.

———. 2011d. “Verification and Validation.” In *A Primer For Model-Based Systems Engineering* (2nd ed.), 98. Blacksburg, VA: Vitech Corporation.

———. 2011e. “What is a Model?” In *A Primer For Model-Based Systems Engineering* (2nd ed.), 31. Blacksburg, VA: Vitech Corporation.

Maier, Mark W., and Eberhardt Rechtin. 2009. *The Art of Systems Architecting* (3rd ed.). Buca Raton, FL: CRC Press.

Merriam-Webster Dictionary. 2013. “Heuristic.” 2013. Accessed 10 September, 2013 http://www.merriam-webster.com/dictionary/heuristic.

Modelio Modeling Solutions. 2013. “SysML Architect.” Modelio Store. Accessed 10 September, 2013 http://www.modeliosoft.com/en/modelio-store/modules/modeling-extensions/sysml-architect.html.

Morris, C. R. and Ferguson, C. H. 1993. “How Architecture Wins Technology Wars.” Harvard Business Review. Accessed 10 September, 2013 http://www.churbuck.com/david/Personal/Standards%20Book/Morris%20and%20Ferguson%20on%20Architecture.pdf.

No Magic, Incorporated. 2013. “Magic Draw.” No Magic, Incorporated. Accessed 10 September, 2013 http://www.nomagic.com/products/magicdraw.html..

Object Management Group. 2012. *OMG Systems Modeling Language v. 1.3.* Object Management Group. Accessed 10 September, 2013 http://www.omg.org/spec/SysML/1.3/.

———. 2010. “SysML and AP233 Mapping Activity.” OMG SysML Portal. Accessed 10 September, 2013 http://www.omgwiki.org/OMGSysML/doku.php?id=sysml-ap233:mapping_between_sysml_and_ap233.

Object Management Group, Sparx Systems. 2007. “MDA Overview.” Sparx Systems. Accessed 10 September, 2013 http://www.omg.org/mda/mda_files/MDA_Tool-Sparx-Systmes.pdf.

Papyrus UML website. 2013. Accessed 10 September, 2013 http://www.papyrusuml.org/.

Sage, Andrew P., and William B. Rouse. 2011. *Handbook of Systems Engineering and Management*. Hoboken, NJ: John Wiley and Sons.

Sparx Systems. 2013. “Enterprise Architect.” Sparx Systems. Accessed 10 September, 2013 http://www.sparxsystems.com/.

Vitech Corporation. 2013. “SysML Modeling.” Vitech Corporation. Accessed 10 September, 2013 http://www.vitechcorp.com/solutions/SysML-Modeling.shtml.

U'Ren, Jim. 2003. “An Overview of AP233 - STEP's Systems Engineering Standard.” Presentation for AP233 Working Group, Defense Technical Information Center 6th Annual System Engineering Conference. 20 October 2003. Accessed 10 September, 2013 http://www.dtic.mil/ndia/2003systems/slides.ppt.

THIS PAGE LEFT INTENTIONALLY BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California